Konsep QoE Layanan Video dan Simulasi Jaringan

Book · December 2014

CITATIONS
0

READS
691

2 authors, including:

Dedy Irawan
Badan Pengkajian dan Penerapan Teknologi
4 PUBLICATIONS **2** CITATIONS

SEE PROFILE

Konsep QoE Layanan Video dan Simulasi Jaringan

A. A. N. Ananda Kusuma
Dedy Irawan

PUSAT TEKNOLOGI INFORMASI DAN KOMUNIKASI
BADAN PENGKAJIAN DAN PENERAPAN TEKNOLOGI
[BPPT]

Kata Pengantar

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas segala bimbingan dan petunjuk-Nya lah buku Konsep QoE (Quality of Experience) Layanan Video dan Simulasi Jaringan ini dapat diterbitkan.

Pusat Teknologi Informasi dan Komunikasi (PTIK) melalui kegiatan Multimedia Digital Networks (MDN) telah memulai kegiatan kerekayasaan terkait kualitas layanan video, penyiapan testbed jaringan, dan simulasi jaringan sejak tahun 2011. Kegiatan kerekayasaan ini didanai oleh DIPA BPPT dan Riset Insentif SINas 2014 dari Kementerian Riset dan Teknologi. Penerbitan buku ini merangkum beberapa aspek dari kegiatan kerekayasaan yang telah dilakukan, dan diharapkan berguna untuk peneliti dan perekayasa, akademisi, dan mahasiswa yang ingin melakukan riset di bidang pengujian kualitas video dengan memanfaatkan tool simulasi jaringan. Buku ini juga berguna untuk pengambil kebijakan di bidang TIK karena konsep QoE masih tergolong baru dan belum mendapatkan porsi yang memadai pada regulasi TIK yang telah ada.

Buku ini terbagi atas dua topik utama, yaitu: kajian konsep pengukuran kualitas video untuk estimasi QoE di sisi pengguna, dan penggunaan tool simulator untuk simulasi pengukuran kualitas video pada jaringan yang disimulasikan. Untuk tiap topik, dipaparkan latar belakang permasalahan, studi pustaka hasil-hasil riset untuk usaha penyelesaian masalah, pemodelan dan simulasi menggunakan tool open-source, dan beberapa uji coba yang telah dilakukan.

Ucapan terima kasih kami sampaikan kepada rekan-rekan perekayasa di Bidang Sistem Komunikasi Multimedia (SKM), anggota Lab Advanced Network Protocol (ANP), dan semua pihak lainnya yang telah membantu penyelesaian buku ini.

Semoga bermanfaat.

Serpong, 18/12/2014

Penulis:

A. A. N. Ananda Kusuma

Dedy Irawan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

Konsep QoS (Quality of Service) dan QoE (Quality of Experience)

I.1 Latar Belakang Keperluan Indikator QoS dan QoE

Tantangan dalam desain dan analisa kinerja jaringan komunikasi multimedia adalah adanya faktor konvergensi yang mengarahkan berbagai bentuk informasi (voice, video, dan data) yang disalurkan melalui infrastruktur jaringan (misalnya berbasis Internet Protocol) dengan konstrain batasan biaya dan reliabilitas. Informasi multimedia itu sendiri disalurkan dari dan ke berbagai jenis pengguna dengan persepsi dan ekspektasi kualitas yang berbeda-beda. Kompleksitas seperti ini memerlukan kehandalan dan kualitas penjaminan, baik yang terukur secara nyata dalam bentuk QoS (Quality of Service) jaringan dan aplikasi, maupun yang terukur secara subyektif atas apa yang dirasakan oleh pengguna dalam bentuk QoE (Quality of Experience).

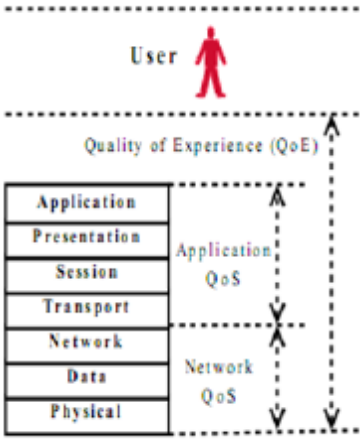
Untuk itu, dalam merancang, mengoperasikan, dan menganalisa kinerja suatu sistem komunikasi multimedia memerlukan pemahaman konsep QoS dan QoE, pengukuran kualitas secara subyektif dan obyektif, penetapan model korelasi QoS/QoE yang sesuai, dan pengembangan simulator dan alat ukur kinerja yang menyertakan model korelasi QoS/QoE. Konsep keilmiahan QoS (aplikasi dan jaringan) sudah tergolong matang sedangan QoE masih digolong dalam tahap awal dan topik yang sedang sangat aktif diteliti. Perbandingan konsep QoS dan QoE dari berbagai sudut pandang dipaparkan pada Bab ini, sebagai pembuka atas berbagai topik yang penulis hendak sampaikan pada buku ini.

I.1.1 QoS (Quality Of Service)

Konsep QoS terkait dengan kualitas layanan yang diberikan oleh suatu infrastruktur jaringan komunikasi. Terkait dengan QoS, terdapat berbagai parameter yang dapat diukur, misalnya bit error rate, delay, jitter, dan sebagainya, yang mana apabila dapat dikelola dengan baik, misalnya dengan me-manage trafik, konfigurasi jaringan yang tepat, diharapkan dapat memberikan jaminan kualitas layanan kepada pengguna infrastruktur jaringan komunikasi ini.

Mekanisme QoS dapat diklasifikasikan menjadi 2, yaitu Application QoS dan Network QoS, sebagaimana ditunjukkan pada Gambar I-1. Untuk Application QoS, sebagai contoh terdapat parameter-parameter QoS seperti resolution, frame rate, video/audio

codec yang terkait dengan aplikasi video, ataupun parameter lainnya untuk aplikasi yang berbeda. Untuk Network QoS, terdapat parameter-parameter QoS seperti packet loss, delay, jitter, dan sebagainya yang umumnya diukur dan dikonfigurasi untuk memaksimalkan kinerja transport pada jaringan. Network QoS sendiri dapat diukur pada beberapa layer dari 7 layer OSI, di mana umumnya mencakup Physical Layer, Data Link Layer, dan Network Layer.



Gambar I-1. QoS dan QoE ditinjau dari Model Layering.

I.1.2 QoE (Quality Of Experience)

Konsep QoE masih tergolong baru khususnya pada layanan multimedia, karena disadari adanya kesenjangan antara kualitas layanan yang disediakan oleh operator dengan apa yang dirasakan oleh pemirsa, khususnya kalau jenis pemirsanya sangat bervariasi sehingga persepsi mereka pun akan bervariasi. Dari Gambar I-1 terlihat bahwa terdapat 2 layer tambahan untuk QoE, yaitu terkait dengan jenis usernya, dan pengalaman dari user, seperti tingkat pendidikan, tingkat sosial, dan aspek-aspek personal non-teknis lainnya.

Terdapat beberapa definisi untuk QoE, di antaranya dari ITU dan Nokia, sebagai berikut:

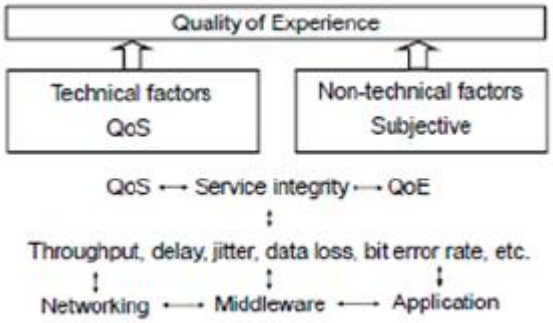
1. ITU-T P.10/G.100 (Vocabulary and Effects of Transmission Parameters on Customer Opinion of Transmission Quality): “The overall acceptability of an application or service, as perceived subjectivity by the end user”

2. Nokia's White Paper: "how a user perceives the usability of a service when in use – how satisfied he or she is with a service"

Beberapa definisi QoE lainnya yang ditampikan pada referensi [1]:

1. "Quality of Experience is the overall performance of a system from the point of view of the users. QoE is a measure of an end-to-end performance levels at the user perspective and an indicator of how well this system meets the user needs"
2. "The user's perceived experience of what is being presented by the Application Layer, where the application layer acts as a user interface front-end that presents the overall result of the individual Quality of Services"

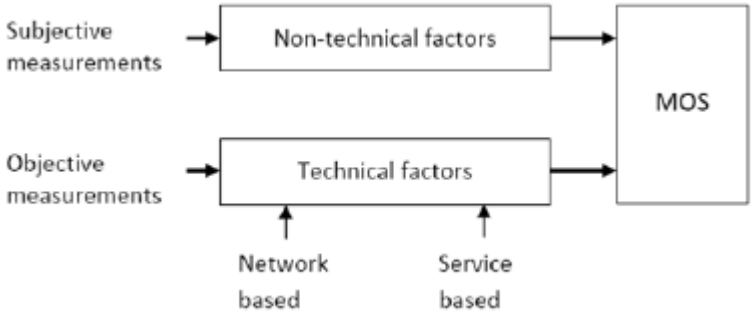
Dari definisi-definisi di atas, walau dinyatakan dengan bahasa yang berbeda, menunjukkan adanya faktor-faktor non-teknis selain faktor-faktor teknis yang menyebabkan berbedanya persepsi pengguna atas suatu layanan. Pengaruh faktor-faktor ini adalah sebagaimana ditunjukkan pada Gambar I-2. Untuk dapat memberikan QoE yang bagus ke pengguna, kedua faktor teknis dan non-teknis harus diperhatikan di segenap layer, baik networking, middleware dan application, yang mana akan saling melengkapi.



Gambar I-2. QoS dan QoE ditinjau dari pengaruh faktor-faktor teknis dan non-teknis.

Walaupun dari definisi konsep QoE ini jelas, namun dalam prakteknya sulit untuk diukur karena persepsi dari pengalaman pengguna yang cenderung bervariasi, dan juga hubungannya dengan parameter-parameter QoS perlu diperjelas. QoE dapat diukur secara langsung dengan meminta pengguna menilai layanan yang kemudian secara statistik dinyatakan dengan suatu nilai MOS (Mean Opinion Score). Pengukuran untuk menentukan MOS ini digolongkan pengukuran secara subyektif yang dianggap telah menyertakan aspek-aspek non-teknis yang berpengaruh.

Secara umum pengukuran QoE dalam bentuk MOS harus mempertimbangkan komponen-komponen pendorong persepsi pengguna, sebagaimana ditunjukkan pada Gambar I-3. Terlihat bahwa skema pengukuran QoE dapat dilakukan secara subyektif maupun obyektif, dan bagaimana hubungan antara pengukuran subyektif dan obyektif memerlukan studi lebih lanjut. Yang menjadi pertanyaan adalah bagaimana pengukuran secara obyektif menggunakan suatu algoritma dapat memprediksi MOS secara akurat. Terdapat beberapa teknik-teknik pengukuran QoE secara obyektif pada pustaka akademis dan industri yang memerlukan kajian tersendiri. Untuk audio QoE dapat dikatakan bahwa teknik-teknik pengukuran QoE relatif sudah matang, namun untuk video QoE status pengukuran secara obyektif masih dalam fase penelitian dan belum ada suatu teknik yang sepenuhnya berlaku secara universal.



Gambar I-3. Komponen-komponen pendorong pengalaman persepsi pengguna.

Selain teknik-teknik pengukuran QoE, hal lain yang perlu dilakukan adalah dalam hal pemahaman QoS mapping (pemetaan parameter-parameter QoS), dan korelasi antara nilai-nilai QoS yang ditawarkan oleh jaringan dan aplikasi dengan nilai-nilai QoE yang dirasakan oleh pengguna.

I.1.3 **Standard-Standard Industri Terkait QoE**

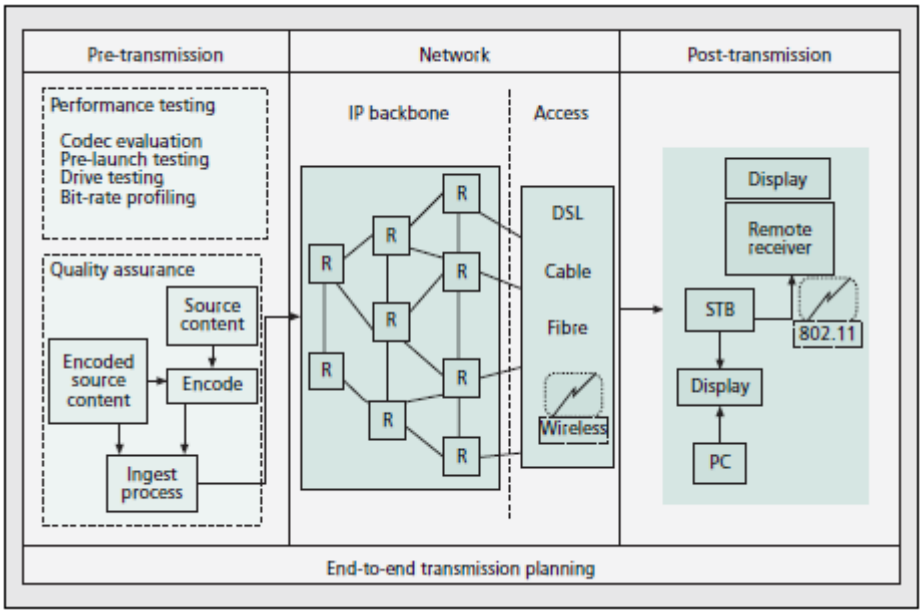
QoE telah didefinisikan oleh berbagai institusi, yang mana pemahamannya masih bersifat umum dan tergantung dengan konteks penggunaannya. Berbagai penelitian terkait dengan QoE menghasilkan standard industri yang dapat dijadikan sebagai referensi bersama untuk berbagai pengukuran QoE. Untuk itu, diperlukan inventarisasi berbagai standard terkait dengan QoE beserta gambaran singkat penggunaannya.

Pengukuran nilai QoE sulit dilakukan karena faktor subyektivitasnya, dan juga karena perlunya data yang besar agar nilainya bermakna secara statistik. Oleh karena itu, umum

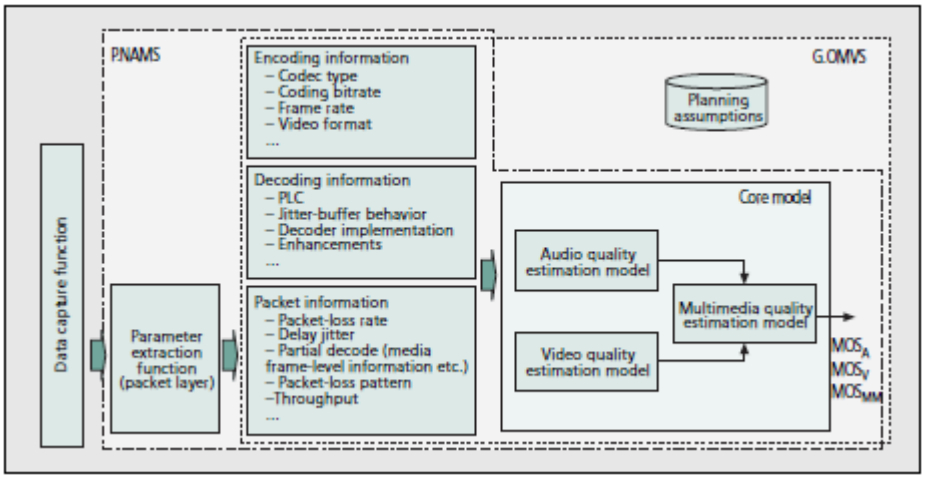
diperlukan suatu teknik kuantitatif yang bersifat obyektif sebagai pendekatan atas teknik subyektif yang dapat dipertanggungjawabkan. Untuk itu, perlu adanya suatu kerangka kerja yang dapat disepakati oleh pihak-pihak yang melakukan pengukuran ataupun pengguna, dan diformalkan dalam bentuk standar-standar industri.

Suatu kerangka kerja pengukuran kualitas pada suatu sistem komunikasi multimedia adalah sebagaimana ditunjukkan pada Gambar I-4 [2]. Untuk memastikan kualitas, pengukuran dilakukan di tiga titik utama, yaitu di Pre-transmission, terkait dengan kualitas konten, pada Network baik core maupun access network, dan di Post-transmission di sisi pelanggan, terkait dengan kualitas receiver, display, dsb. Pada tiap titik-titik pengukuran diperlukan suatu spesifikasi teknis yang distandarisasi. Berbagai spesifikasi teknis untuk assessment (kajian) kualitas dapat diklasifikasikan atas model sebagai berikut.: Media Layer Model, Parametric Packet-Layer Model, Parametric-Planning Model, Bitstream Layer-Model, Hybrid Model. Contoh model berdasarkan parameteric-planning ditunjukkan pada Gambar I-5.

ITU (International Telecommunication Union) – Telecommunication Standardization Sector (ITU-T) Study Group 12 (SG12) telah melakukan investigasi persyaratan dan metode penilaian QoE untuk berbagai layanan multimedia, salah satunya IPTV. Yang lain, misalnya SG9 melakukan untuk cable tv. Selain ITU, terdapat juga VQEG (Video Quality Expert Group) yang tugasnya sebagai tim penasehat teknis, khususnya pada pengujian obyektif untuk kualitas video. Selain itu, ada juga aktivitas yang dilakukan oleh ATIS (Alliance for Telecommunications Industry Solutions) untuk IPTV Interoperability Forum, dan ETSI (European Telecommunications Industry Solutions) untuk menentukan metode-metode pengujian kualitas pada layanan IPTV. Untuk mengkoordinasikan berbagai aktivitas tersebut, ITU membentuk FG (Focus Group), salah satunya FG-IPTV untuk studi QoS/QoE pada IPTV.



Gambar I-4. Kerangka kerja pengukuran kualitas [2].



Gambar I-5. Kerangka kerja pengukuran berdasarkan Parametric Planning [2].

Beberapa standar yang relevan untuk aktivitas pengukuran QoE menggunakan kelima model seperti dipaparkan di atas, adalah sebagai berikut:

1. ITU-T J.148: Requirements for an objective perceptual multimedia quality model.
2. ITU-T J.144: Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference. Pengukuran menggunakan media-layer model. Bersama-sama dengan ITU-R Rec BT.1683 menampilkan 4 buah objective media-layer quality model yang dapat digunakan untuk standard definition television. Sedangkan standard untuk speech, dapat merujuk ke ITU-T P.862.

3. ITU-T G.1070: Opinion model for video-telephony applications. Pengukuran menggunakan parametric planning model, dan menyertakan nilai-nilai koefisien pada fungsi estimasi kualitas yang tergantung pada codec dan packet loss.
4. ITU-T J.246: Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth.
5. ITU-T J.247: Objective perceptual multimedia video quality measurement in the presence of a full reference.
6. ITU-T J.249: Perceptual video quality measurement techniques for digital cable television in the presence of a reduced reference.
7. ITU-T J.342: Objective multimedia video quality measurement of HDTV for digital cable television in the presence of a reduced reference signal

Untuk pengukuran subyektif:

1. ITU-T P.800: Methods for subjective determination of transmission quality
2. ITU-R Rec BT.500-13: Methodology for the subjective assessment of the quality of television pictures
3. ITU-T P.910: Subjective video quality assessment methods for multimedia applications
4. ITU-T P.911: Subjective audiovisual quality assessment methods for multimedia applications

Untuk sinkronisasi audio dan video:

1. ITU-R BT.1359-1: Relative Timing of Sound and Vision for Broadcasting.

I.2 Model Korelasi QoS/QoE

Tantangan untuk analisa kinerja adalah memetakan nilai QoS yang dapat diukur secara obyektif ke nilai QoE yang bergantung ke banyak faktor baik teknis dan non-teknis. Hubungan secara kualitatif antara QoS dan QoE dapat dipahami dari sensitivitas nilai QoE atas perubahan nilai QoS [3] [4].

Mengacu ke Gambar I-6, terlihat tiga wilayah yang mewakili sensitivitas nilai QoE terhadap perubahan nilai QoS, yaitu:

- **Wilayah 1: Constant optimal QoE**

  Pada wilayah ini, persepsi pengguna atas suatu layanan multimedia sudah sangat bagus, sehingga sedikit perubahan nilai QoS tidak akan berdampak.

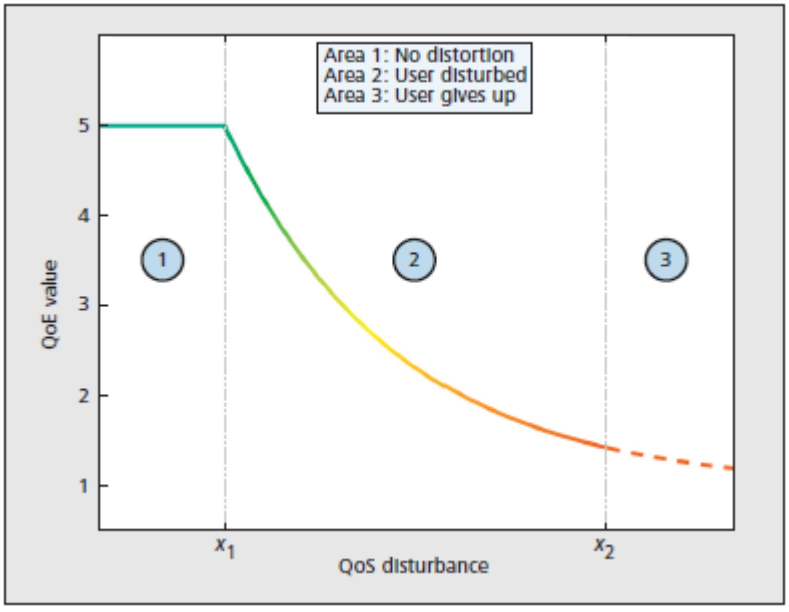
Apabila perubahan nilai QoS sudah mulai diamati oleh persepsi pengguna, maka ini akan masuk ke wilayah 2.

- **Wilayah 2: Sinking QoE**

  Pada wilayah ini, nilai QoE sangat sensitif atas perubahan nilai QoS, dan sedikit memburuknya nilai parameter berakibat pada jatuhnya persepsi pengguna. Dampak jatuhnya nilai QoE ini sangat terasa pada yang nilai awal QoE nya sudah tinggi. Dengan kata lain, pengguna yang mempunyai ekspektasi nilai QoE tinggi sangat sensitif atas perubahan nilai QoS, atau analoginya pelanggan yang sudah terbiasa dengan layanan bagus sangat sensitif atas perubahan kualitas. Dengan makin rendahnya nilai QoE, sensitivitas akan berkurang.

- **Wilayah 3: Unacceptable QoE**

  Pada wilayah ini, nilai QoE sudah tidak dapat diterima lagi, dan tidak responsif lagi terhadap perubahan nilai QoS.



Gambar I-6. Kurva pemetaan sensitivitas QoE karena perubahan nilai QoS [3].

Tantangan riset ada pada pemilihan model korelasi yang dapat memetakan nilai QoS ke QoE secara akurat. Dengan ketersediaan model ini, suatu jaringan dapat didesain secara optimal dengan sumber daya yang efisien dengan suatu prediksi keyakinan pengguna puas atas layanan yang diberikan. Terdapat beberapa model korelasi

QoS/QoE yang telah diusulkan pada komunitas akademik [5], dan beberapa yang terkait dengan pengujian kualitas video dipaparkan pada buku ini.

I.2.1 **Model Exponential**

Fungsi untuk menghitung nilai QoE adalah fungsi banyak variabel yang merepresentasikan faktor-faktor teknis dan non-teknis, dan untuk tiap faktor yang mempengaruhi QoS dinyatakan dengan QoS_i, QoE bisa ditulis sebagai:

$$QoE = f(QoS_1, QoS_2, ..., QoS_n) \tag{I.1}$$

Apabila fokus pada satu faktor nilai QoS, maka persamaan dapat dinyatakan dalam bentuk

$$QoE = f(QoS) \tag{I.2}$$

Dari berbagai nilai QoE (antara batasan nilai QoE_{min} dan QoE_{max}) yang dihasilkan oleh suatu nilai QoS, maka dipilih satu target nilai QoE sebagai indikator kepuasan, yaitu:

$$QoE^* = \min\{\max\{QoE, QoE_{\min}\}, QoE_{\max}\} \tag{I.3}$$

Untuk selanjutnya, gunakan notasi QoE untuk target nilai indikator kepuasan ini. Dari hubungan kualitatif di atas, di mana perubahan nilai QoE akibat perubahan nilai QoS tergantung tingkatan dari nilai QoE nya, maka dapat diasumsikan suatu partial differential equation sebagai berikut:

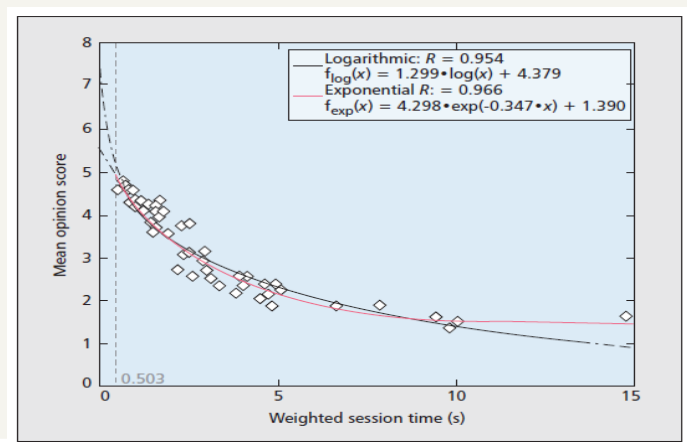
$$\frac{\partial QoE}{\partial QoS} \cong -(QoE - \gamma) \tag{I.4}$$

Dari hubungan kuantitatif pada persamaan (I.4), dapat diturunkan suatu persamaan dengan relasi eksponential sebagai berikut:

$$QoE = \alpha . e^{-\beta . QoS} + \gamma \tag{I.5}$$

Relasi exponential ini adalah suatu hipotesa yang dikenal sebagai IQX (Exponential Interdependency of Quality of Experience and Quality of Service) Hypothesis [3]. Dibandingkan dengan relasi logaritmic, relasi exponential ini lebih mendekati asumsi hubungan kualitatif sebelumnya di mana nilai QoE akan lebih sensitif pada nilai QoE yang tinggi, sedangkan untuk relasi logaritmic grafiknya akan lebih melandai yang berarti kurang sensitif pada nilai QoE tinggi. Contoh korelasi menggunakan hubungan

exponential dan logaritmic pada aktivitas web browsing ditunjukkan pada Gambar I-7, dan tampak relasi exponential lebih representatif pada wilayah yang penting yaitu wilayah 2 (Singking QoE), dan tidak signifikan pada wilayah 3 (Unacceptable QoE).



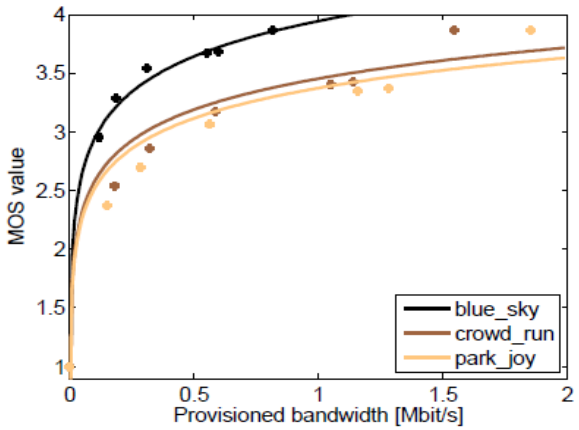
Gambar I-7. Contoh hubungan pemetaan Exponential dan Logaritmic untuk aktivitas Web Browsing [3].

Gambaran di atas menunjukkan superioritas model exponential dibandingkan logaritmic, dan untuk selanjutnya perlu dikaji penggunaanya untuk layanan video.

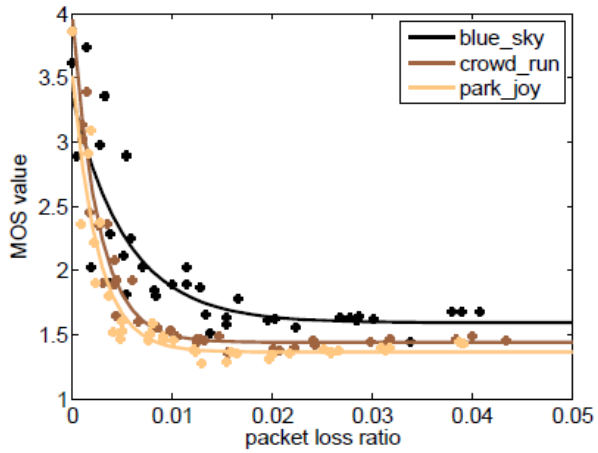
Pada referensi [6], dilakukan eksperimen pada aplikasi video untuk menunjukkan peran relasi exponential QoS dan QoE. Eksperimen dilakukan untuk uncontrolled dan controled distortion dari streaming suatu video clip. Dalam rangka estimasi pengaruh controlled dan uncontrolled video distortion, digunakan metode SSIM sebagai full-reference metric. Untuk pemetaan antara SSIM dan MOS digunakan exponential fitting function $f(x)= 13.91\ e^{1.715.x}$. Dengan exponential function ini, diperoleh rentang MOS antara 1.44 – 3.86.

Controlled reduction artinya terdapat perubahan paramater sumber daya yang disengaja (deliberate) seperti resolusi, frame rate, dan sebagainya. Sedangkan untuk uncontrolled reduction, artinya result yang diperoleh didasarkan random packet loss. Eksperimen yang dilaporkan pada referensi [7] menggunakan 3 buah video clip yang berbeda, yaitu blue_sky, crowd_run, dan park_joy. Sebagai catatan, resource-related satisfaction rating function terkait dengan reduksi kapasitas. Untuk protocols dan applications yang elastic, throughput dapat beradaptasi dengan kondisi jaringan, sehingga controlled adaptation dapat menghasilkan nilai MOS yang tinggi. Di lain pihak, perubahan yang uncontrolled, misalnya disebabkan oleh packet loss, dapat berakibat pada perubahan yang cukup signifikan pada kepuasan pengguna.

Gambaran di atas direfleksikan pada provisioning curve pada Gambar I-8 dan delivery curve padaGambar I-9. Pada Gambar I-8, terlihat bahwa QoE meningkat dengan meningkatnya available bandwidth. Terlihat bahwa dengan available bandwidth sekitar 0.5 Mbps sudah cukup memperoleh persepsi pengguna yang baik untuk ketiga video clip.



Gambar I-8. Provisioning Curves: Hubungan MOS dengan Bandwidth disediakan [6].



Gambar I-9. Delivery Curves: Dampak packet loss ratio terhadap MOS [6].

Untuk dampak packet loss pada uncontrolled distortion ditunjukkan pada Gambar I-9. Terlihat bahwa packet loss yang sangat kecil, kurang dari 1%, berdampak besar pada nilai MOS. Mengkombinasikan hasil-hasil di atas, diperoleh bahwa dampak packet loss terhadap persepsi pengguna lebih besar dari pada controlled bandwidth reduction.

Paparan di atas telah menunjukkan prospek penggunaan model exponential untuk estimasi nilai QoE layanan video. Salah satu standard industri untuk estimasi QoE

layanan multimedia, yaitu standard ITU-T G.1070 (dijelaskan di BAB III dan digunakan pada simulator di BAB VII), juga memiliki pola yang serupa dengan model exponential.

I.3 Model ARCU untuk Analisa Multi-Dimensi QoE

Bagian sebelumnya menunjukkan adanya berbagai faktor, teknis dan non-teknis, yang mempengaruhi persepsi pengguna atas layanan multimedia yang diestimasi dengan nilai parameter QoE atau MOS. Untuk itu perlu dikaji ketersediaan suatu model matematika yang dapat memilah faktor-faktor pengaruh tersebut pada dimensi-dimensi tersendiri, dan bagaimana masing-masing faktor tersebut berkontribusi pada pemetaan nilai QoE. Salah satu model terkini yang melakukan analisa multi-dimensi QoE adalah model ARCU [8], yang dijelaskan sebagai berikut.

I.3.1 Kerangka Kerja Model ARCU (Application-Resource-Context-User)

Model ARCU yang generik didasarkan atas 4 multi-dimensional spaces, yaitu:

1. **Application space (A)**

Meliputi faktor-faktor yang dipengaruhi oleh konfigurasi aplikasi atau layanan, misal encoding, frame rate, buffer size, SNR, content type, dan sebagainya.

2. **Resource space** (R)

Dimensi yang merepresentasikan karakteristik dan kinerja teknis dan sumber daya jaringan dalam penyediaan layanan, misal faktor delay, jitter, loss, dan throughput yang merepresentasikan kinerja jaringan, ataupun kemampuan pemrosesan server dan end-user device yang merepresentasikan kinerja pemrosesan informasi.

3. **Context space** (C)

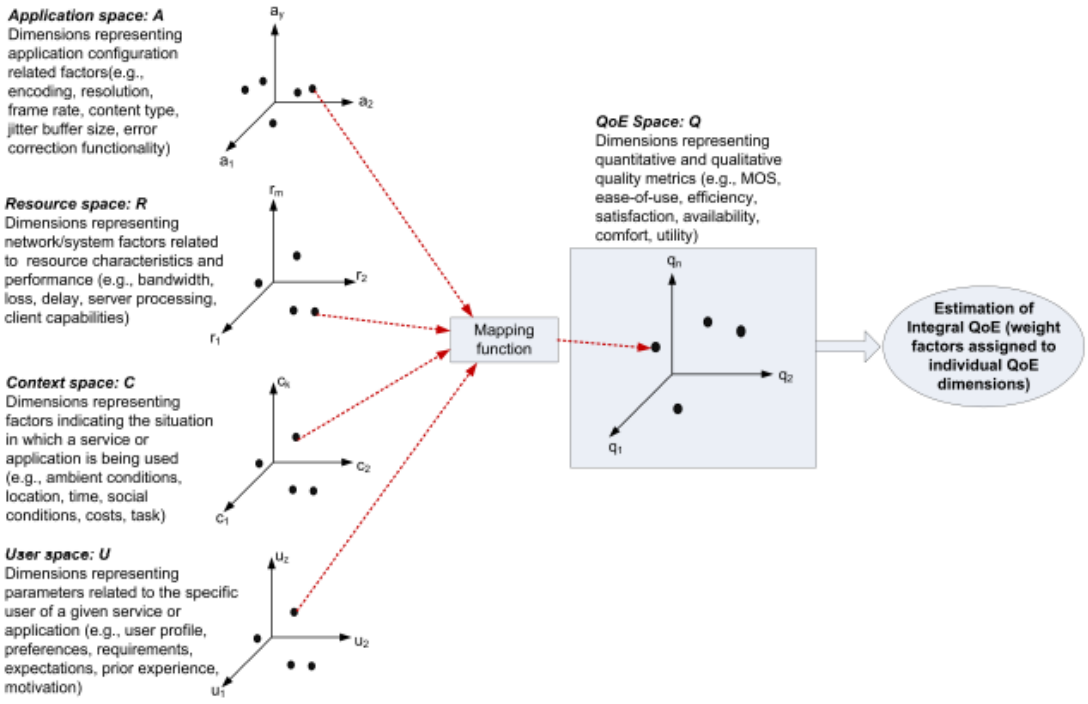
Dimensi yang mengidentifikasikan situasi di mana layanan atau aplikasi digunakan, misal kondisi ambient, lokasi pengguna, waktu, konteks sosial, dan konteks ekonomis seperti services costs.

4. **User space** (U)

Dimensi terkait dengan pengguna spesifik atas layanan atau aplikasi, misal faktor-faktor terkait data demografik, user preference, requirements, expectations, prior knowledge, mood, motivasi, dan sebagainya.

Ilustrasi model ditunjukkan pada Gambar I-10, di mana dimensi untuk tiap space dapat mengacu ke skala yang berbeda-beda. Titik-titik dari ARCU space dipetakan ke titik-titik pada QoE space, yang mencakup dimensi yang merepresentasikan quantitative dan qualitative quality metrics yang berbeda-beda.

Walau secara konsep tiap titik pada tiap space diasumsikan independent, dalam kenyataan sering terjadi korelasi antar himpunan bagian dari parameter yang berbeda, ataupun pada space yang sama, misal loss rate dan delay pada jaringan cenderung berkorelasi. Untuk itu diperlukan suatu fungsi yang mendefinisikan wilayah yang valid pada ARCU space yang telah mempertimbangkan pengaruh korelasi pada situasi riil.



Gambar I-10. Model ARCU (Application-Resource-Context-User) [8].

Mapping function (MF) adalah fungsi pemetaan MF: ARCUv →QoE. Melanjutkan dari pemetaan ke QoE space adalah penentuan nilai integral QoE, sebagai kualitas keseluruhan yang dipengaruhi oleh berbagai faktor dimensi. Evaluasi atas keseluruhan persepsi ini dapat berdasarkan atas weighted (pembobotan), bisa non-linear, kombinasi dari berbagai quality evaluation metrics (dimensions). Hal selanjutnya adalah dalam hal

penentuan bobot yang berpengaruh pada faktor kontribusi tiap metrics ke integral QoE. Jika QoE space terdiri atas dimensi $q_1,\ldots,q_n$ dengan bobot $w_1,\ldots,w_n$, maka

$$QoE_{Integral} = f(w_1 q_1, \ldots, w_n q_n) \tag{I.6}$$

I.3.2 **Contoh Penerapan Model ARCU**

Contoh dimensi pemodelan untuk beberapa layanan ditunjukkan pada Tabel I-1.

Tabel I-1. Contoh dimensi pemodelan untuk berbagai layanan.

Service example	Input Dimensions				QoE dimensions
	A	R	C	U	
Streaming video	codec frame rate resolution playout buffer size content type spatial information index	throughput delay jitter loss rate loss burstiness device CPU device resolution device memory display brightness display color depth	user mobility location lighting conditions service cost	age motivation gender	color quality blurriness jerkiness blockiness
VoIP	codec bit rate FEC echo cancellation loss concealment	throughput delay jitter loss rate loss burstiness audio rendering (headset / handsfree)	ambient noise user mobility service cost language	age motivation gender hearing level	intelligibility clipping talking quality conversational quality listening quality coloration noisiness
Multiplayer gaming	game genre scene update rate dead reckoning algorithms game mechanics cheating protection	throughput delay loss rate device CPU display resolution GPU power device memory	number of players player environment playing task	age gender skill player class motivation	interactivity immersion enjoyment plausibility

Sebagai contoh, untuk one-way streaming service, untuk tiap input dimension ARCU, disertakan faktor-faktor yang berpengaruh. Untuk A space: codec, frame rate, resolution, playout buffer size, content type, spatial information index. Content type dapat merujuk ke perbedaan video sequences pada spatial information dan temporal information indexes. Untuk R space: throughput, delay, jitter, loss rate, loss burstiness, device CPU, device resolution, device memory, display brightness, display color depth. Untuk C space: user mobility, location, lighting conditions, service cost. Untuk U space: age, motivation, gender. Contoh motivasi, apakah pengguna menggunakannya sebagai entertainment atau distance learning akan punya dampak berbeda tentang bagaimana kualitas diestimasi. Untuk dimensi QoE dapat dipilih color quality, blurriness, jerkiness, blockiness. Integral QoE kemudian dihitung dari QoE dimesion ini dengan menggunakan bobot-bobot sesuai dengan penekanannya. Sebagai contoh, ada studi yang

menyimpulkan bahwa meningkatnya video jerkiness berdampak lebih besar ke integral quality dibandingkan dengan blockiness.

I.4 Alat Bantu Visualisasi Pemetaan QoS dan QoE

Secara umum, QoE adalah fungsi pemetaan dari sejumlah tak hingga metrics QoS, sehingga walaupun secara eksplisit dapat diestimasi formulasinya, tetap saja sulit untuk dipahami dan kurang fleksible digunakan untuk analisa kinerja. Salah satu alat bantu yang relatif baru diusulkan pada kalangan akademis untuk tampilan multivariate observation adalah radar chart [9], dan dijelaskan sebagai berikut.

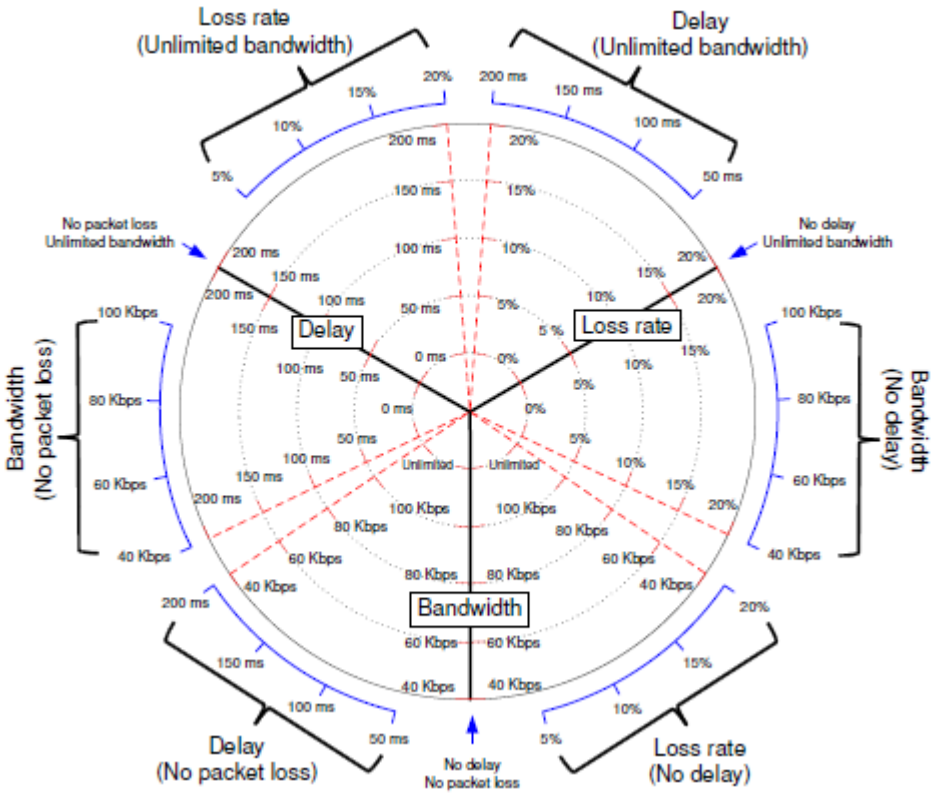
I.4.1 Konsep Radar Chart

Radar chart tergolong tool untuk visualisasi planar, dan contoh layout ditunjukkan pada Gambar I-11, di mana QoE dipetakan sebagai fungsi dari 3 buah QoS metrics. Pada Gambar I-11, radar chart terbagi atas 3 sectors untuk delay, loss rate dan bandwidth, yang masing-masing memiliki 5 arcs. Contoh, untuk sector bandwidth, dari tengah sampai ke pinggir, masih-masing arc mewakili unlimited bandwidth, 100 Kbps, 80 Kbps, 60 Kbps, dan 40 Kbps. Masing-masing sector terbagi atas 2 half-sectors, di mana axis pembaginya memotong semua arcs pada sector. Sebagai contoh, pada bandwidth sector yang bertetangga dengan delay sector, terdapat masing-masing bandwidth-delay half sector, yang axisnya dibatasi oleh no delay dan 200 ms delay pada bandwidth sector, dan unlimited bandwidth dan 40 kbps bandwidth pada delay sector. Tiap perpotongan antara axis dan arc pada radar chart menandakan kondisi jaringan saat QoE dari suatu aplikasi diukur. Centering axis pada sector selalu menandakan situasi metrics yang sempurna, kecuali yang direpesentasikan oleh sector tersebut. Pada radar chart, dapat dibentuk suatu polygon untuk nilai QoE atau MOS tertentu, dengan menghubungkan titik-titik QoS yang membentuknya.

I.4.2 Analisa Kinerja Menggunakan Radar Chart

Ada 2 cara penggunaan radar chart untuk aplikasi analisa kinerja. Yang pertama gunakan untuk aplikasi yang sama dengan menyusun polygon-polygon jika nilai MOS ditingkatkan, dan pengguna radar chart dapat melihat bagaimana respons aplikasi dengan persyaratan nilai QoS yang makin ketat. Cara kedua adalah dengan tampilkan

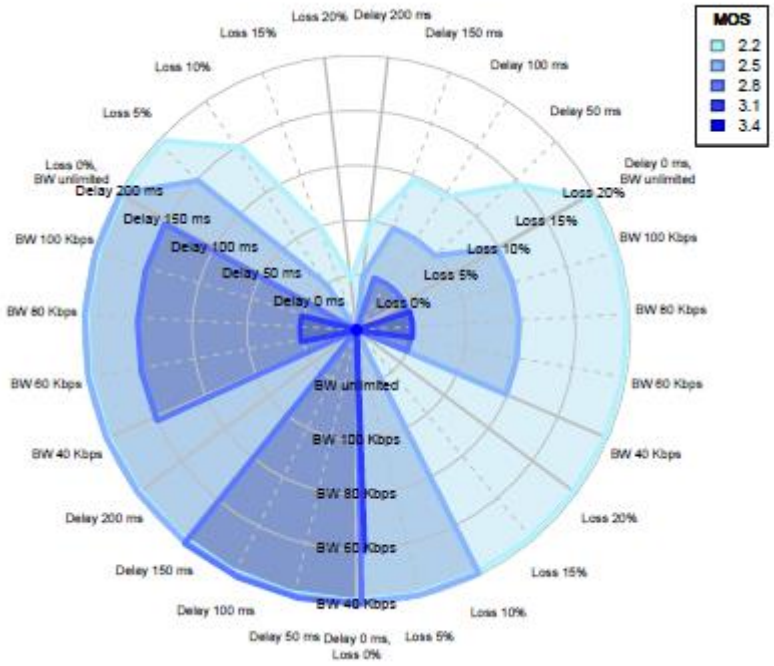
aplikasi-aplikasi berbeda pada radar chart, dan pada nilai MOS yang sama, kelebihan dan kekurangan dari masing-masing aplikasi dapat diidentifikasi.



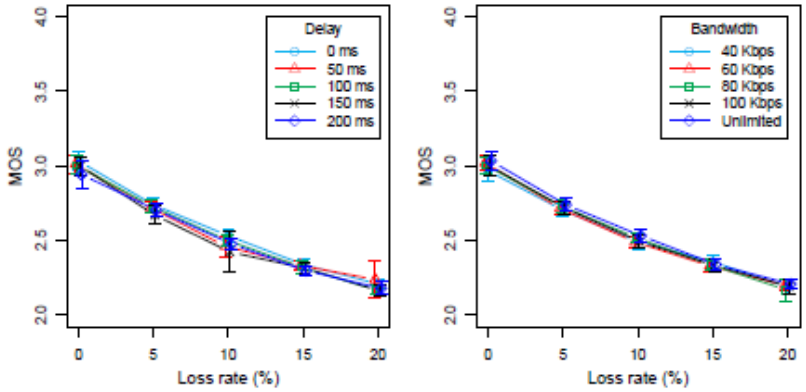
Gambar I-11. Contoh layout Radar Chart [9].

Gambar I-12 menunjukkan contoh analisa pada satu aplikasi, yang menunjukkan karakteristik radar chart dari aplikasi GoogleTalk. Terlihat bahwa polygon-polygon menyusut secara steady (stabil) sampai kira-kira MOS mencapai 3.1, di mana setelah ini GoogleTalk tidak dapat memberikan QoE yang lebih baik, walaupun kondisi network sempurna. Juga terlihat bahwa area sector untuk loss rate menunjukkan penyusutan yang cepat, dilanjutkan dengan kontraksi dari bagian delay pada bandwidth axes dan arcs. Hal ini menunjukkan bahwa software GoogleTalk ini sensitif terhadap packet drops dibandingkan dengan faktor-faktor yang lain.

Fenomena yang diamati pada radar chart ini dapat juga dijelaskan dengan graf tradisional, namun perlu lebih dari satu graf. Contoh ditunjukkan oleh Gambar I-13, yaitu 2 buah graf tentang perubahan MOS terhadap loss rate masing-masing dengan variasi delay dan bandwidth.



Gambar I-12. Contoh karakteristik Google Talk pada Radar Chart [9].



Gambar I-13. MOS (Mean Opinion Score) Google Talk sebagai fungsi dari packet loss rate [9].

BAB II

Pengukuran Kualitas Video

II.1 Metric Pengukuran

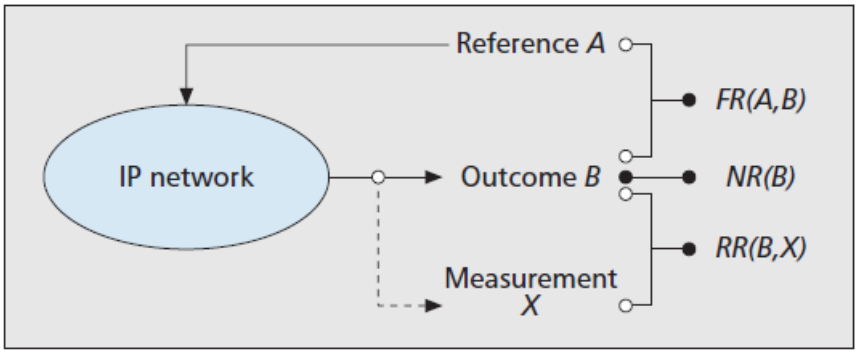
Nilai QoE ditentukan oleh faktor-faktor teknis dan non-teknis, dan oleh karena itu terdapat berbagai metrics pengukuran didasarkan atas ketersediaan data untuk perbandingan. Selain itu, metode untuk pengukurannya juga sangat bervariasi, baik yang sifatnya subyektif maupun obyektif [3] [10] [11].

Dalam melakukan pengukuran diperlukan suatu perbandingan antara [3]:

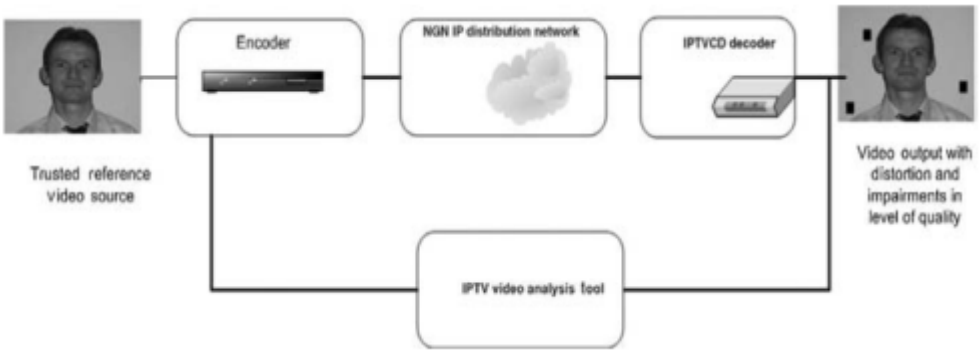
1. **Reference**
   Ini adalah referensi awal berupa konten, baik image, video, audio, yang masih murni (belum terdistorsi), atau bisa juga suatu aktivitas downloading yang tidak mengalami gangguan. Informasi dari reference ini diperlukan saat penentuan rating dari kualitas keluaran.

2. **Outcome**
   Ini adalah hasil keluaran dari suatu transmisi data yang berpotensi dalam bentuk image, video, audio yang telah mengalami distorsi, atau misalnya aktivitas downloading yang mengalami delay. Distorsi atau gangguan yang terjadi pada outcome ini akan berpengaruh pada persepsi pengguna akan kualitas konten. Dengan kata lain, yang tersisa dari konten yang tidak terdistorsi menentukan nilai QoE.

Kedua parameter di atas sifatnya generik dan berlaku untuk berbagai kondisi jaringan dan layanan. Untuk keperluan pengukuran kualitas video, reference dapat berupa file video sumber yang belum terdistorsi (uncompressed), dan outcome dapat berupa file video yang diterima di sisi penerima, yang mungkin saja telah mengalami distorsi karena pengaruh codec ataupun disrupsi pada infrastruktur jaringan. Didasarkan atas ketersedian data dari penyedia konten dan jaringan sebagai reference, metrics pengukuran dibagi menjadi tiga, dengan ilustrasi ditampilkan pada Gambar II-1.



Gambar II-1. Ilustrasi klasifikasi metrics pengukuran untuk QoS-QoE [3].



Gambar II-2. Contoh pengujian QoE yang Full-Reference (FR) pada sistem IPTV [10].

Ketiga macam metrics pengukuran berdasarkan atas ketersediaan reference adalah sebagai berikut:

1. **Full Reference (FR) metrics**

   Pada kasus ini informasi lengkap dari outcome dan reference tersedia untuk perbandingan, sehingga pengujian secara subyektif dan obyektif dapat dilakukan secara detil. Gambar II-2 menunjukkan contoh pengujian dengan FR metrics pada sistem IPTV skala laboratorium, di mana video analysis tool melakukan komparasi antara keluaran video yang mengalami distorsi (outcome) dengan sumber video (reference). Teknik pengukuran PSNR dan SSIM yang dijelaskan di sub-BAB II.2 dan II.3, dan digunakan di BAB VII tergolong metric ini.

2. **No Reference (NR) metrics**

   Pada kasus ini informasi kualitas diekstrak dari outcome dikarenakan tidak tersedianya informasi dari reference. Ini hal yang umum terjadi pada sistem yang online di mana fokus hanya pada persepsi kualitas oleh pengguna atau suatu algoritma yang merepresentasikannya. Teknik pengukuran menggunakan

standard ITU-T G.1070 yang dijelaskan di BAB III dan digunakan di BAB VII tergolong metric ini.

3. **Reduced Reference (RR) metrics**
   Pada kasus ini, reference dan outcome diturunkan dan dibandingkan dari suatu himpunan parameter-parameter yang diukur dari jaringan. Sebagai contoh, QoE dari level aplikasi dapat dideskripsikan dari hybrid image quality metric (HIQM), kemudian QoS dari jaringan direpresentasikan dengan variasi throughput dan losses yang diukur pada jaringan.

Pengujian kualitas dapat dilakukan secara subyektif maupun obyektif, dan masing-masing dibahas sebagai berikut:

II.1.1 Pengujian secara Subyektif

Pengujian secara subyektif dilakukan oleh suatu panel penilai riil atas kualitas suatu layanan. Karena tiap penilai dapat memiliki persepsi yang berbeda-beda, dan juga pada variasi pandangan pada s**u**atu tingkat penilaian yang sama, maka pengujian secara subyektif ini harus dilakukan untuk jumlah sampel penilaian dari jumlah pengguna yang besar, agar hasil-hasil pengujian dapat dijustifikasi secara statistik. Sehingga pengujian secara subyektif ini cenderung memakan waktu lama dan biaya yang besar.

Umumnya mekanisme pengujian secara subyektif ini telah didefinisikan secara formal pada standard-standard seperti ITU, yang didasarkan atas hasil-hasil pengujian yang telah mereka lakukan. Nilai dari suatu pengujian subyektif didefinisikan sebagai MOS (Mean Opinion Score). Untuk tiap sampel data penilai memberikan nilai antara 1 sampai 5 menurut persepsi kualitas yang dirasakan. Detil metode untuk melakukan pengujian subyektif untuk memperoleh nilai MOS dijelaskan pada [12]. Contoh hasil penilaian berdasarkan tingkatan kualitas pada layanan IPTV diringkas pada Tabel II-1, yang mana skala penilaian ini juga serupa untuk layanan-layanan multimedia lainnya.

Penilaian yang lebih spesifik pada aspek-aspek tertentu dari layanan juga dapat disusun nilai MOS nya, misalnya:

1. MOS-V : nilai untuk kualitas video
2. MOS-A : nilai untuk kualitas audio
3. MOS-AV : nilai keseluruhan dari kualitas audio dan video
4. MOS-C : nilai untuk kualitas interaksi dengan IPTV stream, misalnya nilai persepsi kepuasan saat menggunakan EPG dan mengganti channel.

Tabel II-1. Contoh nilai MOS untuk pengukuran kualitas IPTV [10].

Perception of IPTV Channel	MOS Score
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

II.1.2 **Pengujian secara Obyektif**

Pada pengujian secara obyektif, pengguna riil digantikan dengan suatu algoritma, yang berusaha memprediksi persepsi dari pengguna riil atas suatu kualitas berdasarkan informasi (sifat-sifat) dari Reference dan/atau Outcome. Beberapa metode pengujian kualitas video secara obyektif, dan berperan untuk estimasi QoE layanan IPTV antara lain:

1. **PSNR (Peak Signal-to-Noise Ratio)**
   Metric ini adalah ratio antara power dari video signal dengan noise dalam decibels. Informasi ini dapat digunakan untuk identifikasi gangguan pada video frame dan menghasilkan nilai yang merepresentasikan QoE dari pengguna.

2. **MPQM (Moving Pictures Quality Metric)**
   Metric untuk mengkaji kualitas MPEG video stream. Metric ini telah menyertakan teknologi yang mereplika pengamatan manusia dan memberikan penilaian pada skala 1 sampai 5.

3. **SSIM (Structural Similarity)**
   Digunakan untuk mengukur similarity (kemiripan) antara dua images. Teknik ini dirancang untuk memperbaiki metode tradisional seperti PSNR yang terbukti tidak konsisten dengan persepsi dari mata manusia. Pada SSIM, degradasi image didasarkan atas persepsi perubahan informasi struktur image. Pada informasi struktur ini, pixel-pixel memiliki inter-dependencies yang besar kalau secara spatial berdekatan.

4. **MDI (Media Delivery Index)**
   Metric pengukuran ini didefinisikan di RFC 4445, dan digunakan untuk mengukur tingkat kualitas di beberapa titik di jaringan. MDI menyertakan mekanisme scoring

untuk indikasi tingkat kualitas video, dan identifikasi komponen-komponen pada jaringan yang berdampak pada QoE di sisi pengguna. MDI metrics ditampilkan atas dua nilai yaitu DF (Delay Factor) dan Media Loss Rate (MLR), yang dipisah dengan colon.

Tiap metric di atas memiliki kelebihan dan kekurangan, dan memerlukan kajian tersendiri. Dalam hal standardisasi, ITU-T J.144 menyediakan panduan penggunaan objective metric untuk mengkaji kualitas video [13].

Pada Bab ini dijelaskan teknik PSNR dan SSIM sebagai landasan teori sebelum digunakan bersama simulator untuk pengujian kualitas video di BAB VII.

II.2 PSNR (Peak Signal-to-Noise Ratio)

PSNR adalah objective metric tradisional yang sangat umum digunakan, dan penilaiannya didasarkan atas MSE (Mean Square Error) metric. Dengan menimbang nilai luminance *Y* untuk frame-frame original dan yang telah diproses, dan asumsikan frame tersusun atas *M*x*N* pixels, maka:

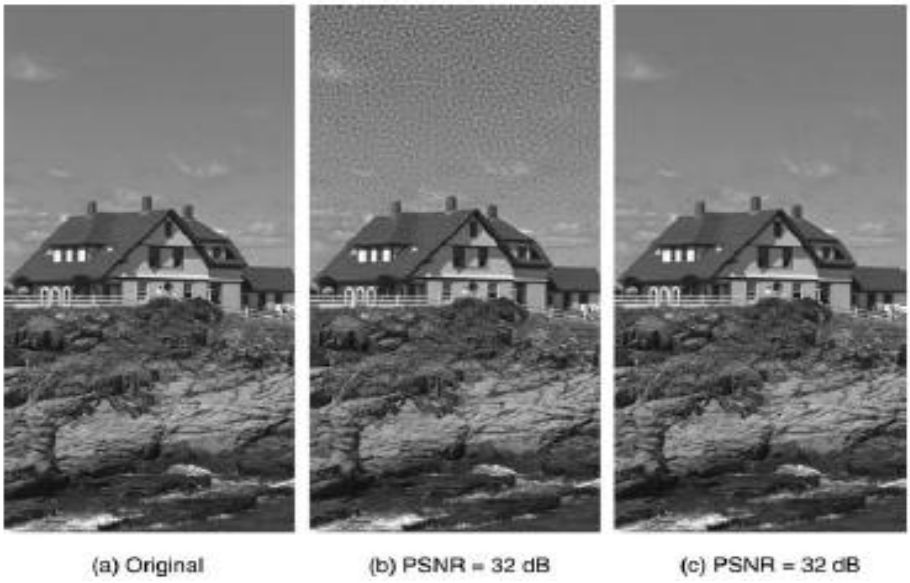
$$MSE = \frac{1}{MxN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left\| Ys(i,j) - Yd(i,j) \right\|^2 \quad \text{(II.1)}$$

$Y_s(i,j)$ dan $Y_d(I,j)$ adalah luminance pada posisi pixel *(i,j)* untuk frame original dan yang telah diproses. Sebagai catatan, di sini didefinisikan untuk luminance, sehingga apabila warna disertakan, maka metric ini menjadi tidak valid. Jika digunakan 8 bits/sample, dan MSE sebagai noise yang menyebabkan perbedaan luminance antara frame original dan yang telah diproses, maka formula PSNR adalah sbb.:

$$PSNR = 20 \log_{10} \left(\frac{255}{\sqrt{\frac{1}{MxN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left\| Ys(i,j) - Yd(i,j) \right\|^2}} \right) \quad \text{(II.2)}$$

Penggunaan pixel-based metrics seperti PSNR dan MSE ini popular karena komputasinya mudah dan cepat. Selain itu, skema untuk meminimalkan MSE ekuivalen dengan least-square optimization yang mana merupakan skema optimisasi yang telah dikenal baik, banyak tool komputasinya telah tersedia, misal [14]. Namun, skema yang

berbasis pixel (sifatnya lokal) dan mengandalkan luminance rentan dengan berbagai efek dari pemrosesan image atau texture dari image. Dapat saja penambahan noise dilakukan dengan sengaja seperti pada image dithering untuk mengurangi error karena quantization colour, sehingga nilai PSNR yang lebih rendah ternyata dapat menghasilkan gambar yang dipersepsikan lebih bagus. Contoh lain karena dampak background dari image, di mana distorsi akan tampak lebih mengganggu pada image dengan latar belakang yang smooth dibandingkan di wilayah gambar yang memiki banyak texture. Dapat disimpulkan bahwa kelemahan dari PSNR adalah diabaikannya hubungan spatial antar pixel yang membentuk image itu sendiri, dan juga kompleksitas dari sistem visual manusia dalam menginterpretasikan suatu image atau perbedaan antara image.



Gambar II-3. Penyertaan nilai noise yang sama pada gambar yang sama [11].

Penggunaan PSNR harus hati-hati karena gambar dengan nilai PSNR yang sama dapat dipersepsikan berbeda oleh mata penilai. Sebagai contoh, pada Gambar II-3 ditunjukkan dua buah gambar dengan nilai PSNR yang sama. Gambar II-3.a adalah image aslinya, dan pada Gambar II-3.b dan Gambar II-3.c adalah hasil dari injeksi sebesar nilai noise yang sama sehingga nilai PSNR pada Gambar II-3.b dan Gambar II-3.c adalah sama. Yang membedakan adalah di wilayah mana dari image, noise tersebut disertakan. Pada Gambar II-3.b, band-pass filtered noise disertakan pada wilayah atas dari image. Pada manusia lebih sensitif terhadap structured (low-frequency) noise dan distorsi ini makin terlihat jelas karena bagian atas image memiliki background yang smooth. Pada Gambar II-3.c, high-frequency noise disertakan pada bagian bawah dari image. Noise yang disertakan pada Gambar II-3.c tidak terlihat karena mata manusia tidak sensitif akan stimulus yang high frequency, dan juga karena ada efek banyaknya

konten texture pada bagian bawah dari image. Hal yang serupa juga terlihat pada komparasi dua buah image pada Gambar II-4.

(a) (b)

Gambar II-4. Gambar dengan nilai PSNR yang sama, memiliki distorsi yang berbeda [15].

II.3 SSIM (Structural Similarity)

SSIM metric dirancang untuk memperbaiki metode tradisional seperti PSNR dan MSE yang tidak konsisten dengan karakteristik HVS (Human Visual System). SSIM metric ini mengukur struktur distorsi atau degradasi dari image yang diharapkan akan memberikan korelasi yang lebih baik terhadap subjective impression. Ide dari struktur informasi adalah adanya ketergatungan spatial dari pixel-pixel, khususnya yang berdekatan. SSIM metric ini didasarkan atas pengukuran dari frame ke frame dengan menggunakan tiga komponen, yaitu luminance similarity, contrast similarity, dan structural similarity, dan menggabungkannya menjadi satu nilai yang disebut index. Index algorithm yang sederhana dan efektif, diajukan oleh Wang [16], selanjutnya dikenal sebagai teknik SSIM adalah sbb.: jika $x = \{x_i \mid i = 1,2,\ldots,N\}$ adalah signal original, dan $y = \{y_i \mid i = 1,2,\ldots,N\}$ adalah signal yang terdistorsi, maka similarity index dihitung sbb.:

$$SSIM = \frac{(2\overline{xy} + C_1)(2\sigma_{xy} + C_2)}{\left[(\overline{x})^2 + (\overline{y})^2 + C_1\right](\sigma_x^2 + \sigma_y^2 + C_2)} \quad \text{(II.3)}$$

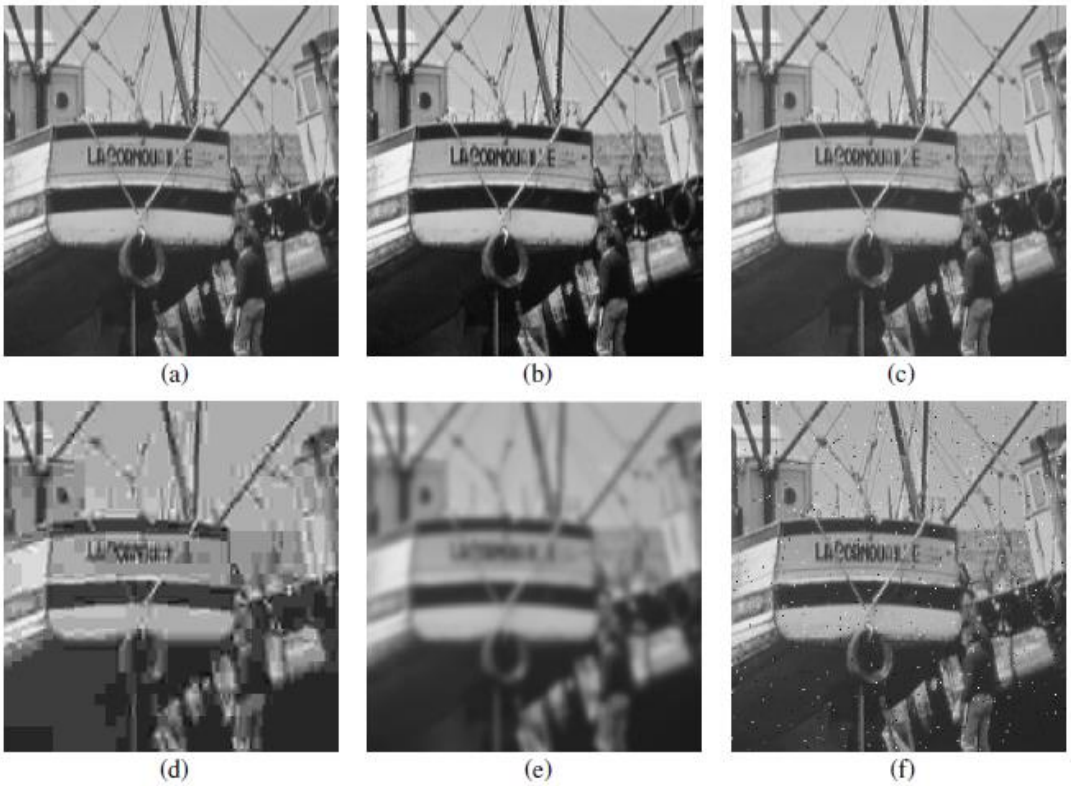
Di mana, $\bar{x}, \bar{y}, \sigma_x, \sigma_y, \sigma_{xy}$ adalah estimasi untuk mean dari *x*, mean dari *y*, variance dari *x*, variance dari *y*, dan covariance *x* dan *y*, sedangkan C_1 dan C_2 adalah konstanta. Nilai dari SSIM adalah antara -1 dan 1, dan nilai terbaik adalah 1 yang menandakan $x_i = y_i$ untuk semua nilai *i*. Index untuk kualitas (quality index) kemudian diterapkan untuk semua image menggunakan sliding window dengan 11x11 circular-symmetric Gaussian weighting function, dan total index adalah nilai rata-rata dari semua quality indexes dari image tersebut. Untuk keseluruhan image, gunakan nilai rata-rata dari SSIM index, dengan notasi MSSIM sebagai berikut:

$$\mathrm{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}(\mathbf{x}_j, \mathbf{y}_j) \tag{II.4}$$

Untuk ilustrasi perbedaan persepsi gambar atas dasar nilai MSSIM, dapat dilihat pada Gambar II-5. Pada Gambar II-5, image asli berupa boat (kapal) ditunjukkan pada bagian (a), dan image ini kemudian diubah dengan berbagai bentuk distorsi, namun dengan MSE (Mean-Square Error) yang sama relatif terhadap image aslinya. Terlihat bahwa walau nilai MSE nya sama, namun persepsi kualitas atas image-image tersebut berbeda. Untuk image-image yang mempertahankan struktur informasinya, seperti bagian (b) dan (c), dengan nilai MSSIM di atas 0.9 terlihat kualitas image-image tersebut tergolong bagus. Dengan struktur informasi yang masih dipertahankan, informasi original bisa didapatkan kembali (fully recovered) dengan pointwise inverse linear luminance transform yang sederhana.

Di lain pihak, untuk image-image yang di-compressed dengan JPEG, mengalami pengkaburan (blurred), atau penyertaan noise pada titik-titik tertentu (penyertaan salt-pepper impulse noise), terdapat struktur informasi yang hilang secara permanent, sehingga sulit untuk mendapatkan kembali kualitas seperti image yang original. Jenis-jenis image seperti ini ditunjukkan pada bagian (d), (e), dan (f) dari Gambar II-5, dan terlihat dengan nilai MSSIM antara 0.6 dan 0.7, image-image ini terlihat buruk dan sulit untuk dipahami kontennya. Terlihat kualitas yang sangat berbeda antara image-image dengan MSSIM sekitar 0.9 dengan image-image dengan MSSIM sekitar 0.6 - 0.7, walau nilai MSE mereka sama.

Dari hasil-hasil pengujian yang telah dilakukan untuk mengkorelasikan nilai-nilai PSNR dan SSIM ke persepsi pengguna dalam format MOS (Mean Opinion Score), diperoleh pemetaan pada Tabel II-2.



Gambar II-5. Perbandingan Image dengan berbagi tipe distorsi [16].

Tabel II-2 Pemetaan PSNR dan SSIM ke MOS (Mean Opinion Score) [17].

MOS	PSNR	SSIM
5 (excellent)	≥ 45	> 0.99
4 (good)	≥ 33 & < 45	≥ 0.95 & < 0.99
3 (fair)	≥ 27.4 & < 33	≥ 0.88 & < 0.95
2 (poor)	≥ 18.7 & < 27.4	≥ 0.5 & < 0.88
1 (bad)	< 18.7	< 0.5

BAB III

Estimasi MOS (Mean Opinion Score) Untuk Kuantifikasi QoE

III.1 Standard ITU-T G.1070 untuk Pengukuran MOS

Seperti telah dijelaskan di Bab sebelumnya bahwa diperlukan model korelasi yang memetakan nilai QoS ke QoE sebagai persepsi kepuasan pengguna atas layanan multimedia. Bab ini menjelaskan penggunaan standard ITU-T G.1070 [18] untuk pengukuran MOS (Mean Opinion Score) sebagai nilai QoE dari pengguna layanan komunikasi multimedia. Standard ITU-T G.1070 merupakan formulasi yang diturunkan dari model matematika empiris yang dikemukakan oleh Yamagishi dan rekan-rekannya dari NTT DoComo untuk pengukuran MOS pada komunikasi multimedia [19]. Model matematika ini menggunakan parameter-parameter obyektif untuk mendapatkan prediksi nilai persepsi subyektif. Model ini tergolong model parametric planning, dan ideal digunakan untuk perancangan sistem dan jaringan multimedia. Kerangka kerja pengukuran ini ditunjukkan pada Gambar III-1.

Buku ini hanya membahas sebagian dari kerangka kerja itu, yaitu hanya fokus pada estimasi kualitas video. Perhitungan kualitas video dipengaruhi oleh faktor-faktor seperti coding distortion, packet loss robustness factor, dan koefisien-koefisien dari hasil pengamatan yang telah tersimpan pada database. Kerangka pengukuran kualitas ditunjukkan pada Gambar III-2, di mana variasi implementasi codec terefleksikan dari nilai-nilai koefisien dari tabel database implementasi codec. Untuk kasus di mana data pengamatan belum ada, maka pengguna perlu melakukan pengukuran sendiri mengikuti aturan yang telah disusun standard ITU-T G.1070, menurunkan nilai-nilai koefisien yang diperlukan, dan memasukkannya ke database untuk digunakan oleh pengukuran lainnya dalam kasus yang sama. Pembaca dapat juga merujuk ke referensi [20] [21] untuk memodifikasi formulasi ITU-T G.1070 dalam adopsi perhitungan menggunakan resolusi atau codec yang berbeda dengan yang tersedia pada standard ITU-T G.1070.

Model ini diturunkan berdasarkan asumsi karakteristik monitor pada Tabel III-1. Untuk kasus yang berbeda, pengguna model ini perlu memberikan catatan-catatan atas deviasi hasil yang diperoleh.

Untuk perhitungan, diperlukan masukan parameter-parameter sbb.:

1. **Video delay (T_v [ms])**.
   Ini merujuk ke end-to-end delay video, artinya telah memperhitungkan processing delay dan jitter-buffer delay. Nilai T_v harus kurang dari 1000 ms.

2. **Spesifikasi Video Codec**
   Koefisien pada model, seperti ditunjukkan pada Gambar III-2, untuk coding dan packet loss distortion diperoleh dengan lookup dari coefficient database.

3. **Tipe Codec dan implementasi**
   Informasi ini diperlukan untuk identifikasi implementasi spesifik dari video codec untuk memperoleh koefisien yang tepat pada perhitungan.

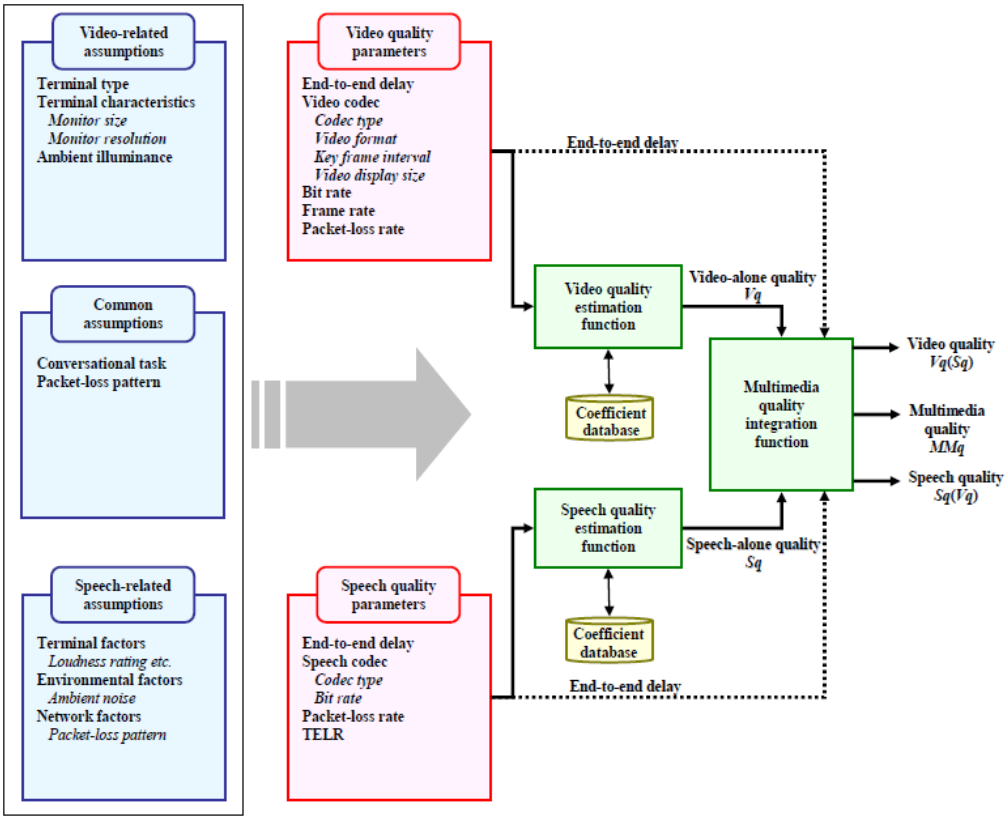
4. **Resolusi spatial**
   Model ini mendukung ukuran video antara QQVGA dan VGA.

5. **Key frame interval**
   Nilai ini menunjukkan interval waktu di mana video di-coded hanya dari intra-frame information. Nilai ini berpengaruh pada efektivitas video coding (yaitu quality versus video bit rate) dan robustness terhadap packet-loss degradation.

6. **Video packet loss rate (Ppl_v [%])**
   Ini merujuk ke end-to-end IP packet loss rate pada video. Hal ini telah menyertakan packet loss pada terminal-jitter buffer dan packet loss pada jaringan. Nilai sebaiknya lebih rendah dari 10%.

7. **Video frame rate (Fr_v [fps])**
   Ini merujuk ke frame rate yang digunakan pada encoder, dan tidak merefleksikan frame repetition yang digunakan oleh decoder. Standard ini mengasumsikan rentang frame rate antara 1 sampai 30 fps.

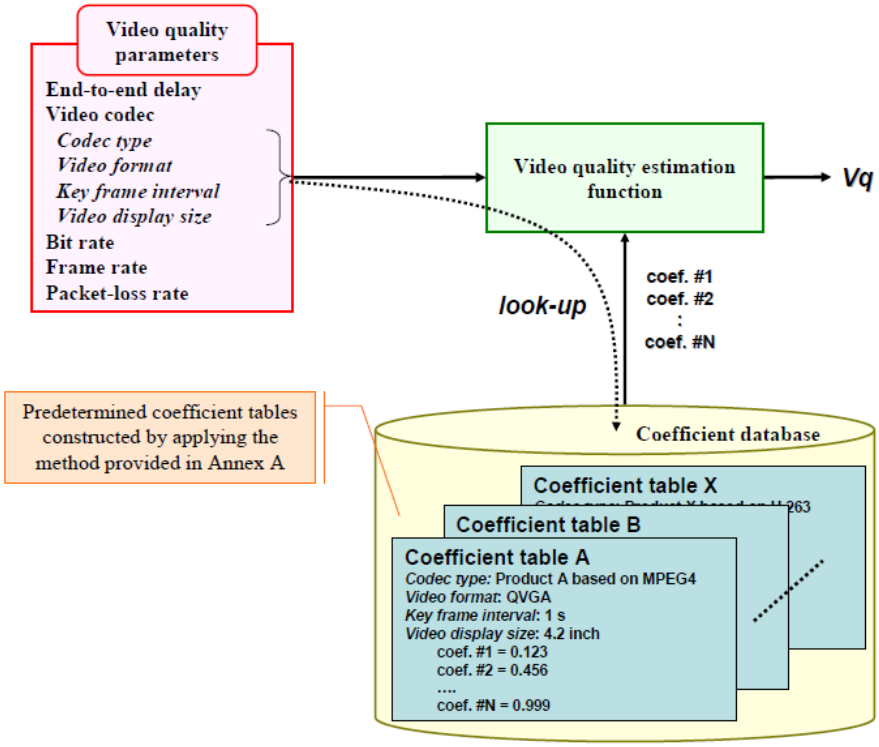
8. **Video bit rate (Br_v [kbit/s])**
   Ini merujuk ke video bit rate pada encoder.



Gambar III-1. Kerangka Kerja Model Kajian Kualitas (*Quality Assessment Model*).

Tabel III-1. Asumsi terkait karakteristik monitor.

Monitor specifications	Value
Diagonal length (Note)	2-10 inches
Dot pitch	<0.30
Colour temperature	6500 K
Bit depth	8 bits/colour
Refresh rate	≥60 Hz
Brightness	100-300 cd/m^2
NOTE – "Diagonal length" refers to the image size on the monitor screen.	



Gambar III-2. Penentuan nilai-nilai koefisien yang tergantung kepada implementasi codec.

III.1.1 Fungsi Estimasi Kualitas Video

Keluaran pada model ITU-T G.1070 adalah multimedia quality (MM_q), speech quality yang tergantung pada video quality ($S_q(V_q)$), dan video quality yang tergantung pada speech quality ($V_q(S_q)$). Penentuan $S_q(V_q)$ dan $V_q(S_q)$ masih dikaji lebih lanjut. Oleh karena itu, fokus pada buku ini hanya pada V_q, yang ditentukan oleh fungsi estimasi kualitas video.

Formula-formula untuk estimasi kualitas video adalah sbb.:

1. **Perhitungan Video Quality (V_q)**.

   V_q dihitung berdasarkan masukan parameter kualitas video sbb.:

$$V_q = 1 + I_{coding} \exp\left(-\frac{Ppl_V}{D_{PplV}}\right) \quad \text{(III.1)}$$

Di mana I_{coding} adalah basic video quality yang dipengaruhi oleh coding distortion (yang dipengaruhi oleh video bit rate (B_{rv} [kbit/s] dan video frame rate (Fr_v [fps]) dan packet loss robustness factor Dp_{plV} yang mengekspresikan degree of video quality robustness karena packet loss (P_{plv} [%]).

2. **Basic Video Quality Dipengaruhi oleh Coding Disrtotion (I_{coding})**

$$I_{coding} = I_{Ofr} \exp\left\{-\frac{\left(\ln(Fr_V)-\ln(O_{fr})\right)^2}{2D_{FrV}{}^2}\right\} \quad \text{(III.2)}$$

Di mana O_{fr} adalah optimal frame rate yang memaksimalkan video quality pada tiap video bit rate (Br_v) dan dinyatakan sbb.:

$$O_{fr} = v_1 + v_2\ Br_V, \quad 1 \le O_{fr} \le 30,\ v_1 \text{ and } v_2\text{: const} \quad \text{(III.3)}$$

Di mana jika $Fr_v = O_{fr}$, maka $I_{coding} = I_{Ofr}$, yang mana I_{Ofr} merepresentasikan maximum video quality pada tiap video bit rate (Br_v) yang dinyatakan sbb.:

$$I_{Ofr} = v_3 - \frac{v_3}{1+\left(\frac{Br_V}{v_4}\right)^{v_5}}, \quad 0 \le I_{Ofr} \le 4,\ v_3, v_4, \text{ and } v_5\text{: const} \quad \text{(III.4)}$$

Di mana D_{FrV} merepresentasikan degree of video quality robustness disebabkan oleh frame rate (Fr_v), dan dinyatakan sbb.:

$$D_{FrV} = v_6 + v_7\ Br_V, \quad 0 < D_{FrV},\ v_6 \text{ and } v_7\text{: const} \quad \text{(III.5)}$$

Di mana koefisien $v_1, v_2, \ldots$, dan v_7 tergantung kepada tipe codec, video format, key frame interval dan video display size, dan diperoleh dari coefficient lookup database.

3. **Packet Loss Robustness Factor (D_{PplV})**

$$D_{PplV} = v_{10} + v_{11}\exp\left(-\frac{Fr_V}{v_8}\right) + v_{12}\exp\left(-\frac{Br_V}{v_9}\right), \quad 0 < D_{PplV} \quad \text{(III.6)}$$

Di mana Ppl_v adalah packet loss rate., dan koefisien v_8,v_9,…, dan v_{12} tergantung kepada tipe codec, video format, key frame interval dan video display size, dan diperoleh dari coefficient lookup database. Pada buku ini, untuk pembahasan di BAB VII, digunakan codec H.264 yang mengacu ke kolom no 5 pada tabel koefisien standard ITU-T G.1070 [18].

III.1.2 **Akurasi dari Model**

Pearson product-moment correlation antara empirical subjective quality dan quality estimate yang dihasilkan oleh model digunakan untuk mengevaluasi akurasi dari fungsi estimasi kualitas video. Nilai korelasi *r* dihitung dari semua test data sets, sbb.:

$$r = \frac{\sum_{v=1}^{V}(y_v - \bar{y})(x_v - \bar{x})}{\sqrt{\sum_{v=1}^{V}(y_v - \bar{y})^2 \sum_{v=1}^{V}(x_v - \bar{x})^2}} \quad \text{(III.7)}$$

Di mana *V* adalah jumlah dari test data sets, dan mean values dari data sets dihitung sbb.:

$$\bar{x} = \frac{1}{V}\sum_{v=1}^{V} x_v \quad \text{(III.8)}$$

dan

$$\bar{y} = \frac{1}{V}\sum_{v=1}^{V} y_v \quad \text{(III.9)}$$

Di mana x_v merepresentasikan estimated quality of test data dan y_v merepresentasikan subjective quality of test data.

Akurasi yang diperoleh model ini telah diverifikasi dengan menggunakan subjective quality database yang menggunakan ITU-T H.264 dan MPEG-4 codecs. Juga koefisien v_1,v_2,… dan v_{12} telah dioptimisasi untuk tiap database. Nilai cross-correlation secara rata-rata menunjukkan nilai 0.975. Validitas nilai optimal koefisien v_1,v_2,… dan v_{12} juga telah divalidasi dengan menerapkannya pada unknown data, dan diperoleh nilai cross-

correlation rata-rata sama dengan 0.955. Detil mekanisme perhitungan, dataset, dan database yang digunakan dapat dirujuk pada dokumen ITU terkait.

III.2 Metode Penurunan Koefisien Empiris pada Standard ITU-T G.1070

Bagian ini melaporkan metode penyusunan koefisien empiris pada standard ITU-T G.1070 dan perencanaan lookup database. Terdapat 12 koefisien yang harus diturunkan dengan cara fitting fungsi berdasarkan masukan parameter yang diberikan, dan dijelaskan sebagai berikut.

III.2.1 Metodologi untuk Menurunkan Koefisien v_1, v_2, …, dan v_7

Menggunakan subjective video quality MOS, dinyatakan dengan V_{qs}, beserta berbagai kondisi nilai video bit rate (Br_v) dan video frame-rate (Fr_v), koefisien v_1, v_2,…, dan v_7 dihitung dengan 4 langkah berikut:

III.2.1.1 *Perhitungan Nilai I_{Ofr}, O_{fr}, dan D_{fr}*

Dengan menggunakan M nilai frame rate berbeda untuk tiap kondisi video rate b_n, Tabel III-2 diperoleh.

Tabel III-2. Hubungan antara Br_v, Fr_v, dan V_q.

Br_V	Fr_V	V_q
b_n	f_1	$V_{qs}(b_n,f_1)$
b_n	f_2	$V_{qs}(b_n,f_2)$
...	...	...
b_n	f_m	$V_{qs}(b_n,f_m)$
...	...	...
b_n	f_M	$V_{qs}(b_n,f_M)$

NOTE 1 – M represents the number of frame rate conditions. $f_1 > f_2 > ..... > f_M$.

NOTE 2 – $V_{qs}(b_n, f_m)$ represents the MOS under the condition with a video bit rate of b_n and a frame rate of f_m.

Dengan menerapkan dataset pada Tabel III-2 pada formula III.1 dan turunannya, nilai-nilai O_{fr}, I_{Ofr} dan D_{fr} dapat di-approximated pada tiap video bit rate, didasarkan atas least-square approximation, dengan hasil yang ditunjukkan pada Tabel III-3.

Tabel III-3. Hubungan antara Br_v, I_{Ofr}, O_{fr}, dan D_{fr}.

Br_V	O_{fr}	I_{Ofr}	D_{fr}
b_1	O_1	I_1	D_1
b_2	O_2	I_2	D_2
...	...	...	...
b_n	O_n	I_n	D_n
...	...	...	...
b_N	O_N	I_N	D_N

NOTE 3 – N represents the number of video bit-rate conditions. $b_1 > b_2 > ..... > b_N$.

III.2.1.2 *Perhitungan Koefisien v_1 dan v_2*

Dengan menerapkan b_n dan O_n untuk $n = 1,2,…,N$ pada Tabel III-3 ke formula III.3, koefisien v_1 dan v_2 di-approximated dengan least-square approximation.

III.2.1.3 *Perhitungan Koefisien v_3, v_4, dan v_5*

Dengan menerapkan b_n dan I_n untuk $n = 1,2,…,N$ pada Tabel III-3 ke formula III.4 koefisien v_3, v_4, dan v_5 di-approximated dengan least-square approximation.

III.2.1.4 *Perhitungan Koefisien v_6, dan v_7*

Dengan menerapkan b_n dan D_n untuk $n = 1,2,\ldots,N$ pada Tabel III-3 ke formula 11-5 koefisien v_6, dan v_7 di-aproximated dengan least-square approximation.

III.2.2 Metodologi untuk Menurunkan Koefisien v_8, v_9,…, dan v_{12}

Dengan menggunakan subjective video quality (V_{qs}) terkait dengan video bit rate (Br_v), video frame rate (Fr_v) dan video packet-loss rate (Ppl_v), koefisien v_8, v_9,…, dan v_{12} dihitung dengan 4 langkah berikut:

III.2.2.1 *Perhitungan Nilai D_{PplV}*

Dengan menggunakan I_{coding} yang dihitung berdasarkan nilai-nilai koefisien yang telah diturunkan di atas ke formula III.2, dan gunakan subjective video quality (V_{qs}) pada formula III.1, packet loss robustness factor D_{PplV} di-approximated dengan least-square approximation dan ditunjukkan pada Tabel III-4.

Tabel III-4. Hubungan antara video bit rate, video frame rate, dan D_{PplV}.

D_{PplV}		Fr_V					
		f_1	f_2		f_m		f_M
Br_V	b_1	D_{b1f1}	D_{b1f2}	…	…	…	D_{b1fM}
	b_2	D_{b2f1}	D_{b2f2}	…	…	…	D_{b2fM}
	…	…	…	…	…	…	…
	b_n	D_{bnf1}	D_{bnf2}	…	D_{bnfm}	…	D_{bnfM}
	…	…	…	…	…	…	…
	b_N	D_{bNf1}	D_{bNf2}	…	…	…	D_{bNfM}
NOTE 1 – N represents the number of video bit-rate conditions. NOTE 2 – M represents the number of video frame-rate conditions. NOTE 3 – D_{bnfm} indicates a temporary value of the packet-loss robustness factor D_{PplV} for a video bit rate of b_n and a frame rate of f_m.							

III.2.2.2 *Perhitungan Koefisien v_8*

Dengan menerapkan f_m dan $D_{PplV} = D_{b1fm}$ untuk $m = 1,2,\ldots,M$ ke formula III.10, koefisien a, b, v_8 dapat di-approximated dengan least-square approximation:

$$D_{PplV} = a + b\exp\left(-\frac{Fr_V}{v_8}\right) \tag{III.10}$$

III.2.2.3 *Perhitungan Koefisien v_9*

Dengan menerapkan b_n dan $D_{PplV} = D_{bnf1}$ untuk $m = 1,2,\ldots,M$ ke formula III.11, koefisien c, d, v_9 dapat di-approximated dengan least-square approximation:

$$D_{PplV} = c + d\exp\left(-\frac{Br_V}{v_9}\right) \tag{III.11}$$

III.2.2.4 *Perhitungan Koefisien v_{10}, v_{11}, dan v_{12}*

Dengan menerapkan v_8, v_9, $D_{PplV} = D_{bnfm}$, $Br_v = b_n$, dan $Fr_v = f_m$ untuk $n = 1,2,\ldots,N$ dan $m = 1,2,\ldots,M$ ke formula III.6, koefisien v_{10}, v_{11}, v_{12} dapat di-approximated dengan least-square approximation.

III.2.3 **Contoh Coefficient Tables untuk Beberapa Codecs**

Penentuan koefisien pada fungsi estimasi kualitas video tergantung pada implementasi dan setting dari codec yang digunakan. Beberapa faktor yang mempengaruhi diantaranya jenis codec, format video, key frame interval, dan ukuran display dari video. Standard ITU-T G.1070 memberikan 5 buah contoh kondisi dengan faktor yang berbeda sebagaimana ditunjukkan pada Tabel III-5. Kondisi yang menjadi acuan pada buku ini adalah kolom nomor 5 dari tabel tersebut, yang merujuk ke codec ITU-T H.264 [22]. Menggunakan kondisi seperti ditunjukkan pada Tabel III-5, maka diperoleh coefficient table untuk video quality estimation function seperti ditunjukkan pada Tabel III-6.

Tabel III-5. Kondisi dalam menurunkan coefficient tables.

Factors	# 1	# 2	# 3	# 4	# 5
Codec type	MPEG-4	MPEG-4	MPEG-2	MPEG-4	ITU-T H.264
Video format	QVGA	QQVGA	VGA	VGA	VGA
Key frame interval (s)	1	1	1	1	1
Video display size (inch)	4.2	2.1	9.2	9.2	9.2

Tabel III-6. Coefficient table untuk fungsi estimasi kualitas video.

Coefficients	# 1	# 2	# 3	# 4	# 5
v_1	1.431	7.160	4.78	1.182	5.517
v_2	2.228×10^{-2}	2.215×10^{-2}	1.22×10^{-2}	1.11×10^{-2}	1.29×10^{-2}
v_3	3.759	3.461	2.614	4.286	3.459
v_4	184.1	111.9	51.68	607.86	178.53
v_5	1.161	2.091	1.063	1.184	1.02
v_6	1.446	1.382	0.898	2.738	1.15
v_7	3.881×10^{-4}	5.881×10^{-4}	6.923×10^{-4}	-9.98×10^{-4}	3.55×10^{-4}
v_8	2.116	0.8401	0.7846	0.896	0.114
v_9	467.4	113.9	85.15	187.24	513.77
v_{10}	2.736	6.047	1.32	5.212	0.736
v_{11}	15.28	46.87	539.48	254.11	–6.451
v_{12}	4.170	10.87	356.6	268.24	13.684

III.3 Contoh Perhitungan MOS Menggunakan Standard ITU-T G.1070

Sebagaimana telah dijelaskan sebelumnya, perhitungan MOS menggunakan standard ITU-T G.1070 memerlukan informasi bit rate dan frame rate dari coded video source dan packet loss rate. Informasi parameter ini dapat diestimasi dengan melakukan pengukuran langsung dari jaringan riil atau pada jaringan yang disimulasikan pada saat desain atau rekonfigurasi jaringan. Mekanisme estimasi parameter-parameter tersebut dapat merujuk ke usulan ekstensi standard ITU-T G.1070 untuk QoE monitoring tool [23] [24] [25].

Untuk membiasakan diri dengan formulasi dari standard ITU-T G.1070, gunakan template excel yang disediakan oleh ITU-T. Template ini juga berguna saat pemrograman modul simulasi (lihat BAB VII), yaitu untuk validasi hasil perhitungan dari modul ITU-T G.1070 yang dikembangkan. Contoh mengacu pada Tabel III-7, dengan input parameter video bit rate = 2000 kbps, frame rate = 30 fps, packet loss = 0.1%,

diperoleh nilai V_q atau MOS = 3.889. Perhitungan ini menggunakan koefisien-koefisien yang mengacu ke kolom no 5 dari Tabel III-6, yaitu codec ITU-T H.264.

Tabel III-7. Template Perhitungan ITU-T G.1070.

G.1070V2007 (revised by Roger Britt, April 2007) [First version of G.1070, based on G.107V2006]

Speech Quality Parameters	M-Model Inputs (G.1070 Default)			
Mean One-Way Delay	Ts	(0)	0.0	ms
Talker Echo Loudness Rating	TELR	(65)	65.0	dB
Equipment Impairment Factor (see Support Info)	Ies	(0)	0.0	
Packet-loss Robustness Factor	Bpls	(1)	1.0	
Packet-loss Probability	Ppls	(0)	0.0	%
Video Quality Parameters				
Video Delay	Tv	(<1000)	0.0	ms
Bit Rate	Brv		2000.0	kbps
Frame Rate	Frv	(1 to 30)	30.0	fps
Packet-loss rate	Pplv	(<10%)	0.10	%
Select Coefficient Factor number	CF	(1-2)	1	

Speech Quality Index (R)	Q =	93.2
Speech Quality (MOS)	S_q =	4.4
Video Quality (MOS)	V_q =	3.889
Multimedia Quality (MOS)	MM_q =	3.1

Eq #			
	Speech quality estimation funtion, Q		
1	Qr	93.193	
	Impairment due to talker echo, Idte		
2	Idte	0	
3	Re	222.5	dB
4	TERV	71	dB
	Equipment impairment factor, Ie-eff		
5	Ieeff	0	
	Calculation of speech quality, Sq		
6	Sq	4.409150284	
	Calculation of video quality, Vq		
7	Vq	3.888759012	
	Basic video quality affected by coding distort		
8	Icoding	3.187859723	
9	Ofr	30	
10	Iofr	3.187859723	
11	Dfrv	1.86	
	Packet loss robustness factor, Dpplv		
12	Dpplv	1.014993937	
	Calculation of multimedia quality, MMq		
13	MMq	3.085780403	
	Audiovisual quality, MMsv		
14	MMsv	2.958942701	
	Audiovisual delay impairment factor, MMt		
15	MMt	3.915	
16	AD	3.915	
17,18	MS	0	

Provisional Bpl values

Codec	Ie	Bpl
G.723.1+VAD	15	16.12
G.729A+VAD	11	19
GSM-EFR	5	10.03
G.711	0	4.3
G.711+PLC	0	25.14

Video & Multimedia Tables I.1, I.2 & I.3/G.107(

Coefficient Factors	1	2
Codec type	MPEG4	MPEG4
Video format	QVGA	QQVGA
Key frame interval (	1	1
Video display size (	4.2 inch	2.1 inch
v1	5.517E+00	5.517E+00
v2	1.290E-02	1.290E-02
v3	3.459E+00	3.459E+00
v4	1.785E+02	1.785E+02
v5	1.020E+00	1.020E+00
v6	1.150E+00	1.150E+00
v7	3.550E-04	3.550E-04
v8	1.140E-01	1.140E-01
v9	5.138E+02	5.138E+02
v10	7.360E-01	7.360E-01
v11	-6.451E+00	6.451E+00
v12	1.368E+01	1.368E+01
m1	-4.457E-01	-6.966E-01
m2	-6.638E-01	-8.127E-01
m3	4.042E-01	4.562E-01
m4	2.321E+00	3.003E+00
m5	-3.255E-01	-1.638E-01
m6	3.309E-01	3.626E-01
m7	1.494E-01	1.291E-01
m8	5.457E-01	5.456E-01
m9	-3.235E-04	-1.251E-04
m10	3.915E+00	3.763E+00
m11	-1.377E-03	-1.085E-03
m12	0.000E+00	1.465E-02
m13	-1.095E-03	-1.002E-03
m14	0.000E+00	0.000E+00

BAB IV

Pengenalan Simulator NS-3

IV.1 Konsep Simulasi Jaringan

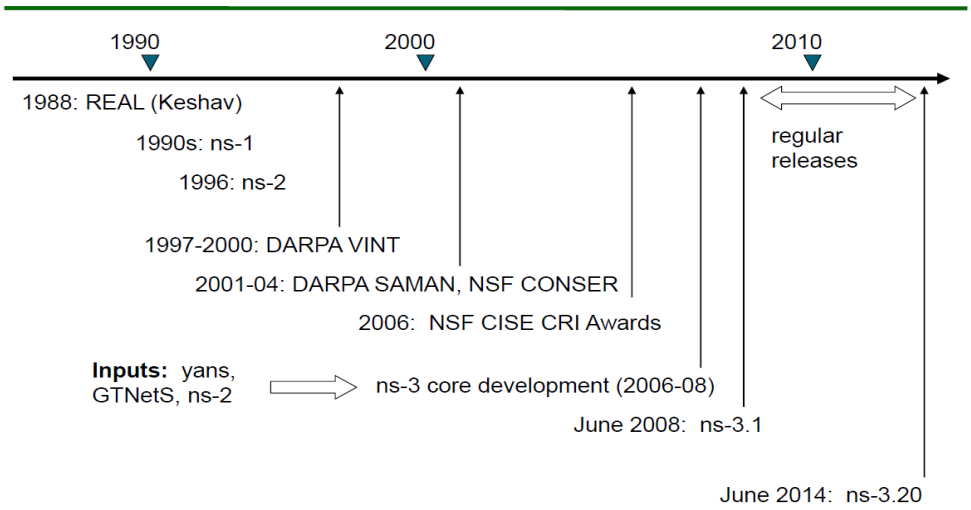
Secara umum terdapat 3 teknik yang berbeda dalam mengevaluasi kinerja dari sistem dan jaringan. Ketiga teknik tersebut adalah analisis matematik, pengukuran, dan simulasi komputer [26]. Semua teknik tersebut masing-masing memiliki kelebihan dan kekurangan. Dalam literatur terdapat beberapa diskusi tentang penggunaan ketiga teknik tersebut, bagaimana cara menerapkannya, dan tingkat kesulitanya dari ketiga teknik tersebut.

Alternatif pilihan dalam mengevaluasi kinerja dari sistem dan jaringan adalah dengan menggunakan sistem real atau membuat sebuah model. Pengukuran membutuhkan ketersediaan sistem riil yang terkadang membutuhkan biaya yang tidak sedikit. Sedangkan analisis matematik dan simulasi komputer menggunakan sebuah model yang merepresentasikan sistem sesuai dengan tujuan yang ingin dicapai. Kedua teknik ini sering digunakan karena tidak membutuhkan biaya besar, dan tidak mengganggu operasional dari sistem riil. Analisis matematik memiliki tingkat kesulitan yang tinggi serta sangat terbatas dalam desain sistem, sedangkan simulasi komputer sering digunakan dalam membandingkan suatu desain model yang berbeda atau mengoptimalisasi desain model dari sistem yang ada. Simulasi komputer sesungguhnya meniru proses sistem setiap waktu [27]. Simulasi komputer dapat diaplikasikan untuk berbagai hal yang berbeda, dan terdapat beberapa jenis simulasi tergantung dengan sistem riil yang dimodelkan, yaitu: *discrete-event simulation, continuous simulation, Monte-Carlo simulation, spreadsheet simulation, trace-driven simulation,* dan sebagainya. Untuk jaringan komputer, teknik simulasi yang dominan adalah *discrete-event simulation.*

Perkembangan perangkat komputasi jaringan yang tumbuh pesat dan semakin kompleks, membuat teknik simulasi jaringan yang akurat dan terukur menjadi hal yang sangat penting. Meskipun dengan mulai munculnya jaringan tesbed dalam skala besar untuk penelitian, teknik simulasi masih memainkan peranan penting dalam hal skalabilitas (baik dari segi ukuran, biaya, dan kecepatan eksperimen), rapid prototyping dan pendidikan [28]. Penelitian yang berbasis pada teknik simulasi diperlukan untuk memperoleh suatu gambaran dari berbagai macam bentuk konfigurasi jaringan dan protocol, tanpa tergantung pada ketersediaan perangkat komputasi fisik. Dalam hal ini, perangkat, aplikasi, data, dan topologi dapat didefinisikan, dan hasil simulasi kemudian dapat dimanfaatkan untuk keperluan analisa jaringan.

IV.2 Peta Jalan NS-3 dan Perbedaannya dengan NS-2

Selama bertahun-tahun perangkat lunak simulator ns-2 menjadi standard *de-facto* [28] bagi kalangan universitas untuk melakukan riset terkait dengan metode komunikasi data dan protocol jaringan. Banyak hasil penelitian jaringan yang dipublikasikan dalam bentuk makalah ilmiah menggunakan simulator ns-2 ataupun modul yang berbasiskan kode sumber ns-2. Meskipun demikian, masih dirasakan perlu adanya tool simulasi baru untuk menjawab berbagai kekurangan yang ada pada ns-2. Beberapa hal yang menjadi catatan adalah: realism (seberapa dekat simulator dengan kebutuhan riil), code-reuse (penggunaan kembali sumber kode), kemudahan untuk debugging, kedekatan bahasa pemrograman dengan bahasa yang digunakan untuk jaringan riil, dan kemudahan untuk software maintenance. Untuk itu, banyak peneliti dan akademisi lainnya membuat sebuah proyek di tahun 2006 dengan melakukan *engineering* (rancang bangun) perangkat lunak simulator jaringan baru pengganti ns-2. Perangkat lunak simulator yang dikembangkan ini adalah ns-3 [29], dengan peta jalan pengembangan ditunjukkan pada Gambar IV-1.



Gambar IV-1. Peta Jalan Network Simulator ns-3 [29].

Beberapa perbedaan antara simulator ns-2 dan ns-3 adalah :

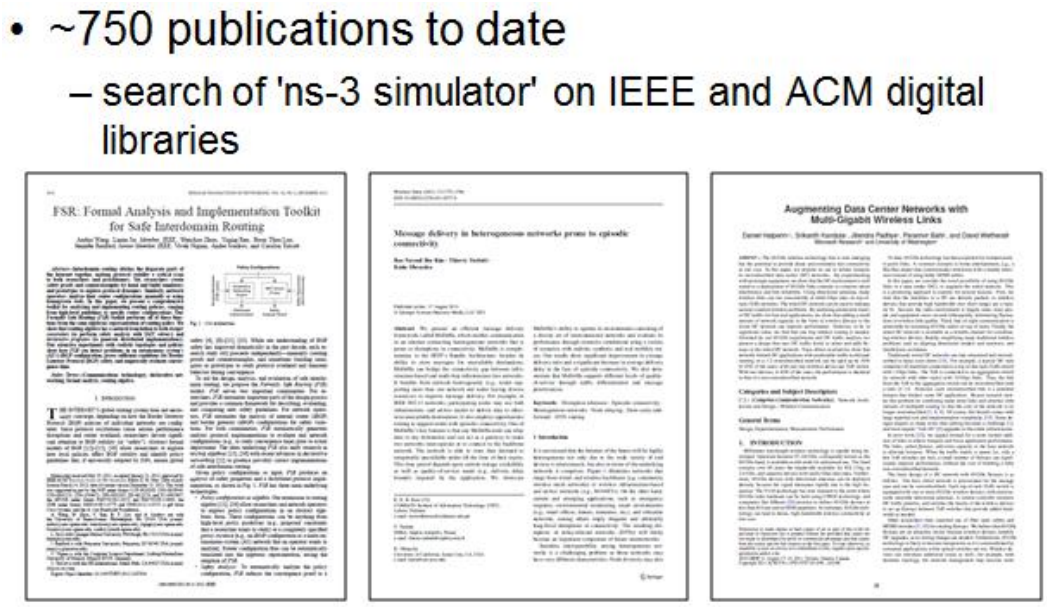
1. Perangkat lunak ns-3 bukan merupakan perluasan dari ns-2. Perangkat lunak simulator ns-3 merupakan perangkat lunak baru. Kedua simulator dibuat menggunakan bahasa C++ tetapi perangkat lunak simulator ns-3 tidak mendukung API ns-2. Beberapa model dari perangkat lunak simulator ns-2 telah diporting ke dalam ns-3 [30].

2. Perangkat lunak simulator ns-3 bersifat open source maka terbuka bagi semua kalangan peneliti dan akademisi atau siapa saja untuk berkontribusi dan membagi program yang telah mereka buat ke dalam ns-3 [30].

Bagi pengguna perangkat lunak simulator ns-2, perubahan yang paling signifikan adalah pada penggunaan bahasa scripting. Program di dalam simulator ns-2 menggunakan bahasa scripting OTcl (versi object oriented dari Tcl). Beberapa komponen simulator ns-2 ditulis dengan menggunakan campuran bahasa antara C++ dan OTcl. Sedangkan dalam simulator ns-3, semua komponen ditulis menggunakan bahasa C++ dengan binding menggunakan bahasa python. Oleh karena itu simulasi dapat dibuat baik menggunakan bahasa C++ atau python.

Belakangan ini popularitas simulator ns-3 makin meningkat, dan merujuk ke IEEE explore dan ACM digital library, sudah terdapat lebih dari 750 publikasi yang dibuat oleh para peneliti dan akademisi menggunakan simulator ns-3 (Gambar IV-2). Laboratorium Advance Network Protokol (Lab ANP) yang merupakan sub unit Bidang Sistem Komunikasi Multimedia pada Badan Pengkajian dan Penerapan Teknologi (BPPT) juga turut aktif dalam menggunakan simulator ns-3. Dari tahun 2012 sampai sekarang, penulis yang menjadi anggota Lab ANP telah melakukan kajian teknologi jaringan menggunakan simulator ns-3. Beberapa karya ilmiah yang dibuat oleh anggota Lab telah dipublikasikan secara nasional.

Simulator ns-3 mungkin belum memiliki semua model yang dimiliki oleh simulator ns-2, akan tetapi seiring pengembangan simulator ns-3 yang masih berlanjut sampai saat ini, sangat dimungkinkan nantinya model di simulator ns-3 akan lebih banyak dari simulator ns-2. Keuntungan lain dari penggunaan simulator ns-3 adalah semakin banyaknya model di simulator ns-2 yang telah diporting ke dalam simulator ns-3, dan juga besarnya dukungan mailing-list pengguna dan aktifitas pengembang dan pengguna simulator ns-3 dibandingkan simulator ns-2.



Gambar IV-2. Jumlah publikasi yang menggunakan simulator ns-3 [29].

.

IV.3 Sumber Daya Pendukung NS-3

IV.3.1 Situs Web

Terdapat beberapa sumber daya yang wajib diketahui oleh pengguna simulator ns-3. Sumber daya simulator ns-3 terdapat di situs resmi http://www.nsam.org. Informasi tersebut dibagi menjadi beberapa sub bagian diantaranya adalah sebagai berikut:

1. *http://www.nsnam.org/overview/what-is-ns-3* memberikan penjelasan singkat tujuan dibuatnya simulator ns-3.
2. *http://www.nsnam.org/releases* memberikan informasi mengenai release dari program pengembangan simulator ns-3 dari versi awal sampai terakhir .
3. *http://www.nsam.org/documentation* memberikan informasi mengenai tutorial, manual sampai API (application programming interface) yang terdapat di simulator ns-3.
4. *http://www.nsnam.org/developers* memberikan informasi mengenai pengembangan simulator ns-3 bagi para developer simulator ns-3.
5. *http://www.nsnam.org/support/faq* memberikan informasi mengenai FAQ (frequently asked question) sebagai bagian dari support simulator ns-3.

IV.3.2 **Mercurial dan Waf**

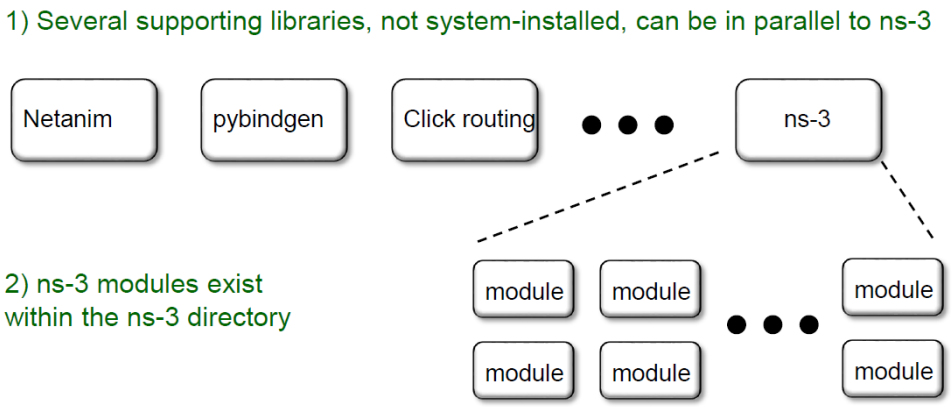
Kode sumber yang kompleks membutuhkan suatu cara untuk mengatur dan mengorganisasikanya. Terdapat banyak perangkat lunak yang dapat melakukan hal tersebut diantaranya yang paling terkenal adalah CVS (concurent version system). Project simulator ns-3 menggunakan mercurial sebagai sistem management bagi kode sumbernya. Mercurial adalah perangkat lunak gratis yang dapat digunakan untuk mengatur dan mengorganisasikan kode sumber. Mercurial dapat menangani berbagai macam kode sumber dari berbagai ukuran, sehingga ideal untuk siapa saja yang bekerja dengan file berversi. Pengguna simulator ns-3 tidak perlu mengetahui terlalu detil tentang mercurial dalam hal untuk mempelajari simulator ns-3. Informasi detil dari perangkat lunak mercurial dapat dilihat pada url *http://www.selenic.com/mercurial*. Perusahaan perangkat lunak Selenic yang merupakan pengembang dari perangkat lunak mercurial menyediakan informasi berupa tutorial dan Quick Start Guide di dalam situsnya.

Waf merupakan kompiler generasi baru menggunakan bahasa Python. Kompiler waf adalah kerangka kerja (framework) berbasis Python untuk melakukan konfigurasi, kompilasi dan installasi aplikasi. Beberapa fitur-fitur penting dari kompiler waf dapat dilihat di situs https://code.google.com/p/waf. Project simulator ns-3 menggunakan kerangka kerja (framework) waf sebagai alat pengembanganya.

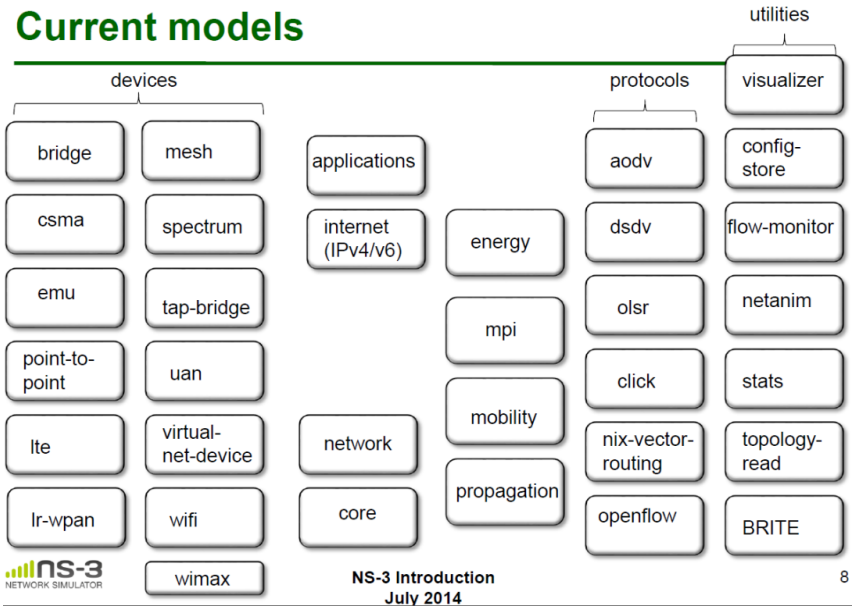
IV.3.3 **Lingkungan Pengembangan**

Pengembangan simulator ns-3 berorientasi pada Unix yaitu berbasis pada command line (konsole) dan ditulis menggunakan bahasa C++ dan Python. Simulator ns-3 diorganisasikan menjadi dua level yaitu perangkat lunak simulator ns-3 itu sendiri dan library eksternal. Pada Gambar IV-3 dapat dilihat pembagian dari kedua level tersebut.

Seperti pada Gambar IV-3, pengembangan pada kerangka kerja simulator ns-3 dibagi menjadi dua bagian yaitu di sisi inti (modul) dalam simulator ns-3 itu sendiri dan di sisi library. Beberapa library yang didukung terletak secara pararel dan tidak terpasang langsung di dalam sistem simulator ns-3 seperti library Netanim, pybindgen, dan Click routing. Sedangkan model yang terbungkus di dalam modul yang digunakan terdapat di dalam direktori simulator ns-3. Simulator ns-3 memiliki banyak model, beberapa diantara model tersebut adalah csma, lte dan wifi. Untuk lebih lengkapnya dari model-model yang ada di dalam simulator ns-3 saat ini dapat dilihat pada Gambar IV-4.

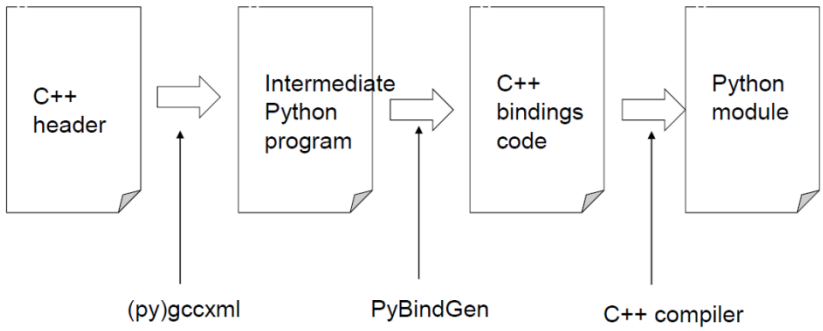


Gambar IV-3. Organisasi perangkat lunak ns-3.



Gambar IV-4. Model pada simulator ns-3.

Pembuatan modul-modul di dalam simulator ns-3 seperti modul bridge, csma, mesh, wifi dan lainnya menggunakan bahasa pemogramman C++ atau Python. Beberapa API (Application Programming Interface) di dalam simulator ns-3 diantaranya ditulis dengan menggunakan bahasa python, tetapi semua model yang masuk kategori inti di dalam simulator ns-3 ditulis menggunakan bahasa C++. Pengguna simulator ns-3 dapat membuat modul menggunakan bahasa python dengan memanfaatkan PyBindGen untuk melakukan semua itu, seperti dapat dilihat pada Gambar IV-5 yang mengilustrasikan proses pembuatan modul menggunakan bahasa python.



Gambar IV-5. Proses binding menggunakan python.

Pengetahuan bahasa pemrograman C++ dan konsep pemrograman berorientasi objek (OOP) mutlak diperlukan dalam menggunakan simulator ns-3. Buku ini tidak menjelaskan secara detil tentang dasar penggunaan bahasa C++, tetapi lebih kepada aspek teknik pemogramman C++ tingkat lanjut. Jika pengguna perangkat lunak simulator ns-3 masih awam menggunakan bahasa C++, disarankan untuk mencari berbagai informasi bisa berupa tutorial dari situs web maupun dari buku cetak. Banyak sekali referensi-referensi pemogramman menggunakan bahasa C++ di Internet dan buku cetak.

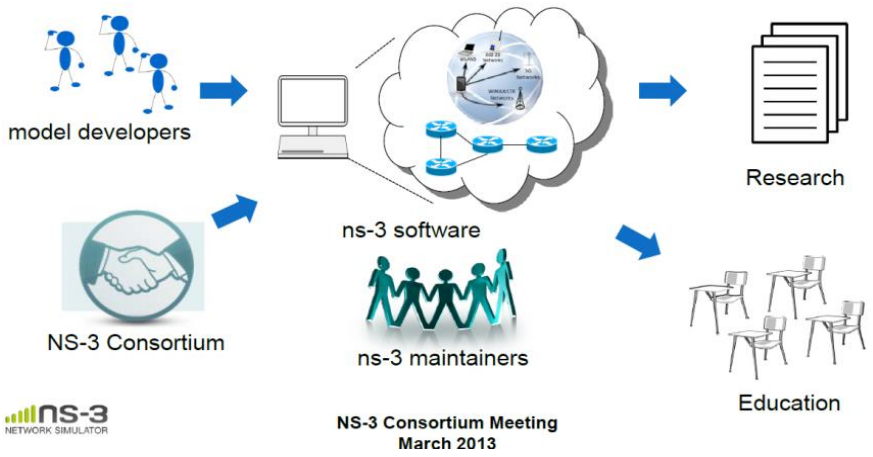
Simulator ns-3 yang memang dirancang agar berjalan pada platform linux, banyak menggunakan beberapa perangkat lunak pengembangan di dalam sistemnya. Beberapa perangkat lunak pengembangan dari GNU (GNU is not Unix) adalah GNU toolchain [31]. GNU toolchain adalah perangkat lunak pengembangan yang disediakan oleh Sistem Operasi GNU/Linux, terdiri dari kumpulan perangkat lunak seperti gcc (compiler C/C++), gdb (GNU Debugger), dan GNU binutils. Sistem ns-3 menggunakan gcc, GNU binutils, dan gdb. Sistem ns-3 tidak menggunakan GNU build system tools (make dan autotools), namun menggunakan kompiler waf untuk build system.

IV.3.4 **Pemrograman Socket**

Project perangkat lunak simulator ns-3 dikembangkan dengan mengikuti arsitektur dari GNU/Linux, yang memiliki interface jaringan (network to device driver) dan interface aplikasi (socket) yang memetakan bagaimana komputer sekarang saling berkomunikasi [28]. Socket yang digunakan oleh perangkat lunak simulator ns-3 adalah API socket berkeley sama seperti yang digunakan pada GNU/Linux. Jika pengguna simulator ns-3 masih baru dalam hal pemrograman menggunakan socket, sangat disarankan untuk mempelajari fungsi API socket dan penerapannya dalam berbagai kasus, khususnya di sistem operasi GNU/Linux. Sebagai catatan, ns-3 juga ditujukan untuk emulasi dan memungkinkan penggunaannya pada testbed dengan device dan aplikasi riil, sehingga kompetensi pemrograman socket untuk komunikasi sangatlah diperlukan.

IV.4 Konseptual NS-3

Simulator ns-3 merupakan simulator jaringan yang bersifat open source dan merupakan *discrete-event simulation* yang intensif digunakan untuk keperluan riset, pengembangan, dan pendidikan. Pada Gambar IV-6 dapat dilihat ilustrasi dari konsep pengembangan simulator ns-3 yang menjelaskan alur proses pengembangannya. Dimulai dari para pengembang model sampai hasil simulator ns-3 yang digunakan untuk keperluan riset dan pendidikan. Pengembang model simulator ns-3 secara umum dibagi menjadi dua yaitu: pengembang model yang tidak terikat (bebas) dan terikat (konsorsium). Biasanya pengembang tak terikat banyak dari kalangan akademisi di universitas dan hobyist. Sedangkan pengembang model yang tergabung pada konsorsium adalah para peneliti yang bekerja di lembaga penelitian dan universitas yang secara khusus melakukan pengembangan model untuk ns-3.



Gambar IV-6. Gambaran umum simulator ns-3.

Pada Gambar IV-6, pengembang model simulator ns-3 membuat model-model yang menggambarkan implementasi dari teknologi jaringan sebenarnya. Simulator ns-3 yang dikembangkan juga di-maintain oleh para maintainer simulator ns-3 dan untuk kemudian didistribusikan untuk keperluan riset dan pendidikan. Seperti kebanyakan perangkat lunak simulasi yang lain, simulator ns-3 juga memiliki pendekatan yang berbeda dalam hal pemodelan, termasuk penggunaan bahasa pemogramman untuk pemodelan dan pemilihan perangkat lunak untuk mengenerate kode sampai pemilihan komponen yang berdasarkan paradigma pemogramman. Pemodelan menggunakan bahasa tingkat tinggi dengan paradigma pemogramman yang dikhususkan untuk simulasi, memiliki keuntungan tertentu. Keuntungan pertama adalah dalam bahasa pemrograman di mana simulator ns-3 memilih menggunakan bahasa C++ dari pada C, karena bahasa C++ memberikan fasilitas yang lebih baik dari pada menggunakan bahasa C.

Keuntungan kedua dari desain simulator ns-3 adalah *reuse* (penggunaan ulang). Sistem ns-3 tidak murni sebagai simulator yang baru, tetapi merupakan *synthesis* (perpaduan) dari beberapa perangkat lunak sebelumnya, termasuk di dalamnya simulator ns-2 (random number generator, pemilihan wireless dan error model sampai routing protocols), Georgia Tech Network Simulator (GTNetS) [32], dan YANS simulator [33] lihat peta jalan ns-3 pada bagian IV.2. Kemudian juga penerapan modul otomasi tabel routing jaringan untuk topologi static yang di porting dari perangkat lunak quagga. Selain itu simulator ns-3 juga memiliki prioritas penggunaan file format input dan output untuk perangkat lunak lain seperti paket trace analyzer, contoh wireshark [34]. Pengguna juga dapat menggabungkan library eksternal seperti GNU Scientific Library untuk keperluan komputasi numerik.

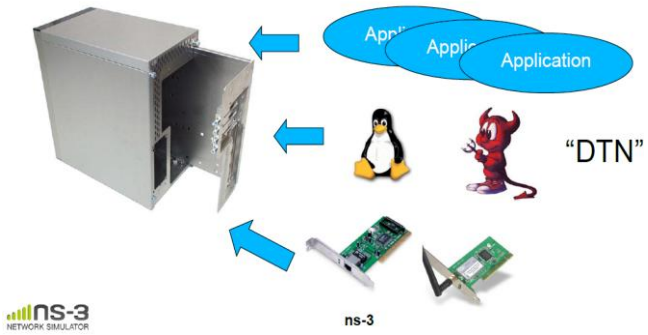
Keuntungan ketiga adalah *ease of debugging* (kemudahan debugging). Arsitektur sistem ns-3 sangat berbeda jauh dari sistem ns-2 yang menggabungkan (mixture) object oriented Tcl dengan C++ sehingga sulit untuk di debug dan penggunaan bahasa Tcl yang kurang familiar bagi beberapa pengguna. Simulator ns-3 lebih menekankan pada kemurnian penggunaan bahasa C++ dan diikuti dengan pemodelan yang lebih ditekankan untuk kinerja dan kemudahan debugging. Sistem ns-3 juga menyediakan API untuk bahasa script python yang membuat simulator ns-3 bisa diintegrasikan ke lingkungan pengembangan berbasis python. Pengguna simulator ns-3 bebas membuat program simulasi baik dengan C++ maupun python. Keuntungan lainnya adalah API low level yang terdapat pada sistem ns-3 dibungkus oleh API yang bernama "helper" yang memberikan kemudahan dalam mengakses API low level di sistem ns-3.

IV.4.1 **Model Elemen-Elemen Jaringan di NS-3**

Hal pertama yang harus diketahui di dalam sistem ns-3 adalah konsep dan abstraksi dari sistem ns-3 itu sendiri. Sebagai tool simulator yang mensimulasikan jaringan secara virtual, simulator ns-3 memiliki semua model dari elemen-elemen jaringan yang meliputi keseluruhan jaringan komputer secara umum.

IV.4.1.1 ***Node***

Di dalam jargon Internet, device komputer yang terhubung ke dalam jaringan disebut sebagai host atau end-system. Karena simulator ns-3 merupakan simulator jaringan dan bukan simulator Internet maka simulator ns-3 tidak menggunakan istilah host, karena istilah host sangat erat kaitannya dengan protocol Internet. Simulator ns-3 menggunakan istilah yang lebih umum yaitu node [30]. Pada Gambar IV-7 dapat dilihat ilustrasi node yang disimulasikan oleh ns-3.



Gambar IV-7. Elemen-elemen jaringan ns-3.

Di dalam simulator ns-3 model node merepresentasikan end-system seperti komputer desktop dan laptop, begitu juga perangkat router, hub, dan switches. Pada sistem ns-3, abstraksi dari model node diterjemahkan ke dalam kelas node. Kelas node memiliki metode untuk membuat node (device komputer). Node merupakan terjemahan dari komputer di dalam simulasi yang memiliki fungsi yang sama seperti komputer nyata di dalamnya terdapat aplikasi, protokol stack dan peripheral cards yang diatur oleh driver untuk menjalankan fungsi tertentu seperti ditunjukan pada Gambar IV-7. Di sini simulator ns-3 menggunakan model yang sama seperti itu.

IV.4.1.2 *Application*

Umumnya, software komputer dibagi menjadi dua bagian yaitu sistem software dan aplikasi. Sistem software mengatur beberapa sumber daya komputer seperti memori, prosessor, hardisk, network dan sebagainya. Sistem software umumnya tidak langsung berinteraksi terhadap pengguna melainkan aplikasi yang berinteraksi secara langsung ke pengguna untuk melaksanakan tugas tertentu.

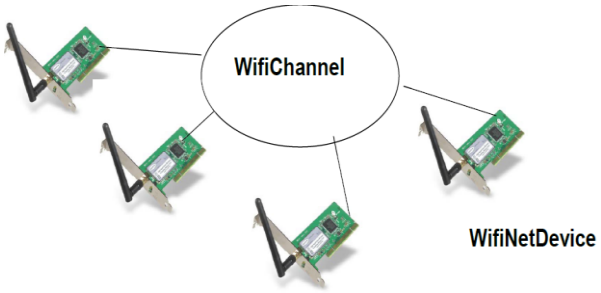
Di dalam sistem operasi riil terdapat batasan yang dibuat untuk memisahkan antara sistem software dan aplikasi yang diatur melalui konsep mode akses. Mode akses ini diatur sedemikian rupa oleh sistem operasi untuk mengendalikan penggunaan sumber daya komputer. Sistem software dapat secara langsung menggunakan sumber daya komputer tetapi aplikasi tidak bisa. Aplikasi membutuhkan persetujuan dari sistem software untuk dapat mengakses sumber daya komputer.

Di dalam simulator ns-3 tidak terdapat batasan terebut, tidak ada konsep sistem operasi dan hak akses atau system call. Di simulator ns-3 hanya terdapat aplikasi yang berjalan di komputer untuk menjalankan fungsi tertentu. Aplikasi dalam ns-3 berjalan di dalam node. Di simulator ns-3 abstraksi program pengguna yang membuat beberapa aktifitas yang disimulasikan dilakukan oleh aplikasi. Abstraksi ini diterjemahkan ke dalam

kelas application, di mana kelas application memiliki metode untuk membuat representasi user level aplikasi di dalam simulasi.

IV.4.1.3 *Net Device dan Channel*

Media komunikasi atau channel merepresentasikan media komunikasi yang digunakan untuk mengirim informasi antara device-device jaringan. Media tersebut bisa berupa fiber optic point-to-point links, shared broadcast-based media seperti ethernet, atau spectrum wireless yang digunakan untuk komunikasi wireless. Abstraksi channel diterjemahkan ke dalam kelas channel. Di dalam kelas channel terdapat beberapa sub kelas channel seperti CsmaChannel yang menterjemahkan teknologi ethernet, PointToPointChannel yang menterjemahkan teknologi fiber optik, dan WifiChannel yang menterjemahkan komunikasi wireless.



Gambar IV-8. NetDevice dan Channel.

Network device merepresentasikan device fisik yang menghubungkan node dengan media komunikasi atau channel. Hal sederhana adalah seperti ethernet network interface card (NIC atau LAN card), atau bisa juga merupakan device wireless IEEE 802.11. Di simulator ns-3 abstraksi dari net device mencakup software driver dan hardware. Net Device dipasang di dalam node agar node dapat berkomunikasi dengan node lain melalui channel. Seperti diilustrasikan oleh Gambar IV-8 abstraksi net device diterjemahkan oleh kelas NetDevice yang berfungsi untuk mengatur koneksi antara node dan channel. Di dalam kelas Netdevice terdapat beberapa sub kelas NetDevice seperti CsmaNetDevice, PointToPointNetDevice, dan WifiNetDevice. Kelas CsmaNetDevice dirancang untuk bekerja dengan CsmaChannel, kelas PointToPointNetDevice dirancang untuk bekerja dengan PointToPointChannel, dan Kelas WifiNetDevice dirancang untuk bekerja dengan WifiChannel.

IV.4.1.4 *Topology Helpers*

Di jaringan riil, mungkin pengguna jaringan komputer dapat memperoleh komputer dengan LAN card yang sudah built-in. Mengikuti analogi riil dengan komputer beserta built-in LAN card, di dalam simulator ns-3 juga terdapat metode untuk membuat node yang terintegrasi dengan net device (built-in). Untuk menghubungkan NetDevices ke Node, NetDevice ke Channels, mengisi IP address, dan lain-lain merupakan pekerjaan manual di dalam simulator. Oleh karena itu simulator ns-3 menyediakan topology helper untuk melakukan semua pekerjaan tersebut sekaligus sehingga memudahkan pengguna simulator ns-3 dalam merancang suatu jaringan. Terdapat beberapa kelas di sistem ns-3 yang berfungsi sebagai helper yang sering digunakan saat eksekusi dan analisis simulasi yaitu:

1. Kelas random variable berfungsi untuk sebagai pembangkit bilangan acak di dalam simulasi. Sebagai contoh, simulasi terhadap perilaku web browser yang dikontrol oleh variable bilangan acak menspesifikasikan think time, rwquest object size, respon object size, dan objek per web page. Lebih jauh, beberapa distribusi di dalam ns-3 dapat digunakan bersama random variable seperti uniform, normal, eksponential, pareto, dan WeiBull.
2. Kelas trace memfasilitasi hasil data logging selama simulasi berjalan, dan dapat digunakan untuk keperluan analisis. objek trace dapat dihubungkan ke setiap elemen network dan membuat informasi dalam format yang berbeda. Format trace yang populer di ns-3 dikenal dengan packet capture log, dikenal dengan pcap. Trace pcap ini dapat divisualisasikan dan di analisa menggunakan beberapa tools analisa yang didesign untuk menganalisa kejadian di network, seperti wireshark dan tcpdump. Simulator ns-3 juga dapat mengenerate trace text sederhana yang dapat merepresentasikan flow packet di dalam simulasi jaringan.
3. Attributes digunakan untuk menkonfigurasi beberapa model elemen jaringan dengan metode set (seperti inisialisasi nilai TTL time-to-live ketika paket IPv4 dibuat). Nilai-nilai dalam attributes dapat diubah melalui command line saat menjalankan simulasi atau dengan mengakses langsung dari API.

IV.4.1.5 *Protocols dan Headers*

Protokol komunikasi dalam hal ini adalah model yang mengimplementasikan deskripsi protokol yang diperoleh dari dokumen standard seperti RFC dan sebagainya, termasuk juga protokol eksperimen terbaru yang belum menjadi standard. Di dalam simulator ns-3 protokol ini diorganisasikan ke dalam protokol stack di mana masing-masing layer di

dalam stack melakukan beberapa fungsi khusus dan terbatas pada paket jaringan, kemudian meneruskan paket ke layer lain untuk diproses kembali.

Protokol header, merupakan subsets dari data yang berada di dalam paket network, dan memiliki format spesifik untuk tiap-tiap protokol. Sebagai contoh, protokol IPv4 dijelaskan dalam RFC 760 yang menjelaskan layout spesifik dari protokol header yang berhubungan dengan IPv4. Beberapa protokol memiliki format standard untuk menyimpan informasi yang berkaitan dengan protokol di dalam paket network.

IV.4.1.6 *Packet*

Network paket adalah fundamental unit dari pertukaran informasi di dalam komputer network. Paket network berisi satu atau lebih protokol header yang menyimpan informasi yang dibutuhkan oleh protokol di end-point dan beberapa router selama perjalanan. Paket umumnya berisi payload yang merepresentasikan data (web page, video dan audio).

BAB V

Penyiapan Platform Simulasi Jaringan Menggunakan NS-3

V.1 Mengunduh Kode Sumber dan Kompilasi

Simulator ns-3 merupakan sistem yang cukup besar dan kompleks, yang memiliki beberapa ketergantungan terhadap komponen lain. Pengguna yang ingin menggunakan simulator ns-3 harus menyiapkan beberapa perpustakaan sistem / library di dalam sistem operasi agar dapat menjalankan simulator ns-3.

V.1.1 Platform dan Kompiler yang didukung

Pada dasarnya simulator ns-3 dikembangkan dengan meniru sistem operasi GNU/Linux. Seperti dijelaskan pada IV.3.3 bahwa simulator ns-3 mengikuti arsitektur GNU/Linux terutama pada arsitektur jaringannya. Oleh karena itu secara default simulator ns-3 dioperasikan pada platform GNU/Linux. Akan tetapi simulator ns-3 juga mendukung beberapa platform yang lain seperti [30] :

1. FreeBSD x86 yang menggunakan kompiler gcc versi 4.2 dan clang versi 3.3
2. FreeBSD x86_64 yang menggunakan kompiler gcc versi 4.2 dan clang versi 3.3
3. Mac OS X Intel yang menggunakan kompiler gcc versi 4.2 dan clang versi 2.79

V.1.2 Dukungan IDE (Integrated Development Enviroment)

Secara default simulator ns-3 tidak menspesifikasikan lingkungan pengembangannya, dan pengguna dapat secara bebas menentukan sendiri lingkungan pengembangannya, baik menggunakan konsole ataupun IDE (Integtared Development Enviroment). Seperti yang telah dijelaskan pada IV.3.3, bahwa pada dasarnya simulator ns-3 dikembangkan dengan mengikuti pola lingkungan pengembangan GNU/Linux yang berbasis konsole. Bagi pengguna yang belum terbiasa dengan itu, maka simulator ns-3 juga mendukung penggunaan melalui IDE dengan tujuan agar lebih memudahkan dalam melakukan pemogramman di dalam simulator ns-3. Beberapa IDE yang di dukung oleh simulator ns-3 adalah:

1. Eclipse (Gambar V-1), merupakan IDE berbasis open-source dan gratis, dan banyak digunakan oleh para pengembang aplikasi. Eclipse mendukung berbagai bahasa pemogramman seperti C++, Java, Python, Perl dan sebagainya.

Gambar V-1. Logo IDE Eclipse.

2. Netbeans (Gambar V-2), merupakan IDE berbasis open-source dan gratis seperti halnya eclipse. Yang membedakan dengan eclipse adalah pada arsitektur pengembanganya. Netbeans mendukung banyak bahasa pemogramman seperti Java, C++, Perl, Python, Ruby, dan sebagainya.

Gambar V-2. Logo IDE netbeans.

3. Qt Framework (Gambar V-3) sudah sejak lama digunakan untuk mengembangkan aplikasi lintas platform. Qt sendiri dibuat pada tahun 1996 oleh perusahaan dari swedia yang bernama Trolltech. Karena sifatnya yang lintas platform, aplikasi dapat dibuat untuk berjalan di atas platform Windows, Linux, dan Mac. Dengan Qt kode yang sama dapat dijalankan pada target platform yang berbeda. Qt Framework sudah didesain sedemikian rupa sehingga mudah digunakan oleh developer tanpa harus mengorbankan fleksibilitas dan efisiensi. Qt mendukung pengembangan dengan bahasa utamanya yaitu Object Oriented C++.

Gambar V-3. Logo Qt Creator.

V.1.3 **Dukungan Terhadap Beberapa Fitur Pilihan**

Pada bagian IV.3.3 sebelumnya telah dijelaskan, bahwa simulator ns-3 terdiri dari 2 tingkat pengembangan yaitu di sisi inti / core dan di sisi perpustakaan sistem / library. Pada sisi perpustakaan sistem / library terdapat beberapa pilihan yang secara default belum aktif atau belum tersedia pada semua platform yang akan dipasang simulator ns-3. Hal ini terjadi karena secara default simulator ns-3 memang tidak menyertakan perpustakaan sistem / library tersebut. Pengguna dapat melihat hal itu ketika melakukan proses konfigurasi. Di akhir proses konfigurasi diperoleh hasil seperti Tabel V-1.

Tabel V-1. Hasil konfigurasi simulator ns-3.

```
---- Summary of optional NS-3 features:
Python Bindings                : not enabled (Python library or
headers missing)
BRITE Integration              : not enabled (BRITE not enabled (see
option --with-brite))
NS-3 Click Integration         : not enabled (nsclick not enabled
(see option --with-nsclick))
GtkConfigStore                 : not enabled (library 'gtk+-2.0 >=
2.12' not found)
XmlIo                          : not enabled (library 'libxml-2.0 >=
2.7' not found)
Threading Primitives           : enabled
Real Time Simulator            : enabled
Emulated Net Device            : not enabled (<netpacket/packet.h>
include not detected)
Network Cradle                 : not enabled (architecture None not
supported)
MPI Support                    : not enabled (option --enable-mpi
not selected)
OpenFlow Integration           : not enabled (Required boost
libraries not found)
SQlite data output             : not enabled (library 'sqlite3' not
found)
Tap Bridge                     : not enabled (<linux/if_tun.h>
include not detected)
PyViz visualizer               : not enabled (Python Bindings are
needed but not enabled)
Use sudo to set bit            : not enabled (option --enable-sudo
not selected)
Build tests                    : not enabled (defaults to disabled)
Build examples                 : not enabled (defaults to disabled)
Scientific Library             : not enabled (GSL not found)
```

Umumnya jika platform belum melengkapi beberapa persyaratan untuk fitur-fitur pada simulator ns-3, akan ditandai dengan keterangan "not enabled". Pada Tabel V-2 dapat dilihat beberapa fitur-fitur default dari simulator ns-3 yang didukung oleh setiap platform.

Tabel V-2. Status option ns-3.

Option	Linux	FreeBSD	Mac OS X
Optimized build	Y	Y	Y
Python bindings	Y	Y	Y
Threading	Y	Y	Y
Real time simulator	Y	Y	N
Emulated net device	Y	N	N
Tap brigde	Y	N	N
Network simulation cradle	Y	?	N
Static builds	Y	Y	Y

V.1.4 **Kebutuhan Paket untuk NS-3**

Simulator ns-3 membutuhkan beberapa persyaratan pada setiap platform, seperti kebutuhan untuk melakukan kompilasi menggunakan gcc, python header dan sebagainya. Daftar di bawah ini menunjukan paket-paket yang dibutuhkan oleh simulator ns-3 yang akan di pasang di Sistem Operasi GNU/Linux. Pada buku ini hanya akan dijelaskan pada distro Ubuntu/Debian, tetapi bisa digunakan pada distro yang lain seperti Fedora/Redhat, Gentoo dan sebagainya.

Ubuntu/Debian

1. Minimal requirement for C++ (release) : minimal set paket yang dibutuhkan untuk menjalankan ns-3 dari tarball.
2. Minimal requirement for Python (release) : minimal set paket yang dibutuhkan untuk python binding dari tarball.
3. qt4 development tools (qt4, bukan qt5) : dibutuhkan oleh network animator.
4. Mercurial dibutuhkan untuk bekerja menggunakan repositori.
5. Menjalankan python bindings dari ns-3 development tree (ns-3-dev) perlu bazaar.
6. Debugging.
7. GNU scientific library (GSL) mendukung error model pada Wifi.
8. The Network Simulation Cradle (nsc) requires the flex lexical analyzer and bison parser generator.
9. Dukungan untuk membaca hasil trace paket pcap.

10. Dukungan database untuk framework statistik.
11. Xml-based version of the config store (requires libxml2 >= version 2.7).
12. GTK-based configuration system.
13. Untuk eksperimen dengan virtual machines dan ns-3.
14. Dukungan untuk utils/check-style.py code style check program.
15. Doxygen dan inline documentation terkait.
16. Ns-3 manual dan tutorial ditulis dalam reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures digambar dengan dia.
17. Dukungan untuk Gustavo Carneiro's ns-3-pyviz visualizer.
18. Dukungan untuk openflow module (mensyaratkan beberapa boost libraries).

Tabel V-3. Instalasi paket ns-3 pada ubuntu 12.04 LTS.

```
apt-get install gcc g++ python
apt-get install gcc g++ python python-dev
apt-get install qt4-dev-tools
apt-get install mercurial
apt-get install bzr
apt-get install gdb valgrind
apt-get install gsl-bin libgsl0-dev libgsl0ldbl
apt-get install flex bison libfl-dev
apt-get install tcpdump
apt-get install sqlite sqlite3 libsqlite3-dev
apt-get install libxml2 libxml2-dev
apt-get install libgtk2.0-0 libgtk2.0-dev
apt-get install vtun lxc
apt-get install uncrustify
apt-get install doxygen graphviz imagemagick
apt-get install texlive texlive-extra-utils texlive-latex-extra
apt-get install python-sphinx dia
apt-get install python-pygraphviz python-kiwi python-pygoocanvas
libgoocanvas-dev
apt-get install libboost-signals-dev libboost-filesystem-dev
apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev
```

V.1.5 **Mengunduh Kode Sumber NS-3**

Simulator ns-3 dapat dipasang dengan menggunakan dua cara yaitu cara otomatis dan manual. Pada buku ini akan dibahas kompilasi dan pemasangan menggunakan kedua cara tersebut. Walaupun buku ini menjelaskan teknik kompilasi dan pemasangan dengan kedua cara diatas, tetapi buku ini lebih menekankan pada penggunaan manual karena nanti pada bab-bab berikutnya akan dibahas tentang bagaimana cara memasang modul baru pada simulator ns-3.

V.1.5.1 *Mengunduh NS-3 Menggunakan Mercurial (Otomatis)*

Kode sumber simulator ns-3 tersedia pada repositori mercurial di server http://www.code.nsam.org. Buat direktori repos di home direktori dengan perintah mkdir (make directory) dan disambung dengan menjalankan perintah hg. Perhatikan Tabel V-4.

Tabel V-4. Mengunduh kode sumber simulator ns-3 cara otomatis.

```
ns-3@ubuntu:~/Public$ mkdir repos
ns-3@ubuntu:~/Public$ cd repos/
ns-3@ubuntu:~/Public/repos$ hg clone http://code.nsnam.org/ns-3-
allinone
```

Perintah `hg clone` berfungsi untuk mengambil kode sumber simulator ns-3 di dalam server ns-3. Pada Tabel V-5 dapat dilihat hasil unduh kode sumber simulator ns-3. Perlu diketahui bahwa proses unduh menggunakan perintah `hg clone` ini akan selalu mengambil kode sumber development dari release simulator ns-3. Jadi pengguna yang ingin menggunakan simulator ns-3 release terakhir dapat menggunakan perintah `hg clone`.

Tabel V-5. Hasil unduh kode sumber simulator ns-3.

```
destination directory: ns-3-allinone
requesting all changes
adding changesets
adding manifests
adding file changes
added 57 changesets with 86 changes to 7 files
updating to branch default
7 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Setelah perintah hg clone selesai, akan muncul direktori ns-3-allinone. Lihat pada Tabel V-6 isi dari direktori tersebut.

Tabel V-6. Direktori ns-3-allinone.

```
build.py  constants.py  dist.py  download.py  README  util.py
```

Terdapat 5 buah file dengan ekstensi python dan 1 buah file README. File-file tersebut yaitu `build.py`, `constants.py`, `dist.py`, `dowload.py`, `README`, dan `util.py`. Pilih file `download.py` untuk mengunduh kode sumber beserta perpustakaan sistem yang dibutuhkan oleh simulator ns-3 dengan menjalankan perintah seperti pada Tabel V-7.

Tabel V-7. Menjalankan file download.py.

```
ns-3@ubuntu:~/Public/repos/ns-3-allinone$ ./download.py
    # Get NS-3
Cloning ns-3 branch
 =>  hg clone http://code.nsnam.org/ns-3-dev ns-3-dev
2998 files updated, 0 files merged, 0 files removed, 0 files
unresolved
    # Get PyBindGen
Required pybindgen version:  0.17.0.886
Trying to fetch pybindgen; this will fail if no network connection
is available.  Hit Ctrl-C to skip.
 =>  bzr checkout -rrevno:886 https://launchpad.net/pybindgen
pybindgen
 *** Did not fetch pybindgen; python bindings will not be available
    # Get NetAnim
Required NetAnim version:  netanim-3.105
Retrieving NetAnim from http://code.nsnam.org/netanim
 =>  hg clone http://code.nsnam.org/netanim netanim
196 files updated, 0 files merged, 0 files removed, 0 files
unresolved
    # Get bake
Retrieving bake from http://code.nsnam.org/bake
 =>  hg clone http://code.nsnam.org/bake
45 files updated, 0 files merged, 0 files removed, 0 files
unresolved
```

Pada Tabel V-7 pengguna simulator ns-3 dapat memperoleh informasi mengenai proses unduh kode sumber ns-3. File `download.py` tidak hanya mengunduh kode sumber ns-3 saja melainkan mengunduh pula beberapa perpustakaan sistem atau modul seperti `PyBindgen, NetAnim dan bake`.

V.1.5.2 *Mengunduh NS-3 Menggunakan Tarball (Manual)*

Proses mengunduh menggunakan tarball sangat berbeda dengan proses menggunakan mercurial. Pengguna simulator ns-3 harus memasukan release dari kode sumber ns-3, unduh, dan kemudian melakukan ekstrak paket. Untuk melihat bagaimana cara melakukan proses unduh menggunakan tarball, perhatikan Tabel V-8.

Tabel V-8. Mengunduh kode sumber simulator ns-3 cara manual.

```
ns-3@ubuntu:~/Public$ mkdir tarball
ns-3@ubuntu:~/Public$ cd tarball/
ns-3@ubuntu:~/Public/repos$ wget http://www.nsnam.org/releases/ns-
allinone_3.21.tar.bz2
```

Buat direktori tarball menggunakan `mkdir` lalu jadikan direktori tersebut sebagai direktori kerja dan jalankan perintah `wget`. Setelah proses unduh menggunakan perintah

`wget` selesai, ekstrak paket `ns-3-allinone-3.21.tar.bz2` dengan perintah `tar -xjvf ns-3-allinone-3.21.tar.bz2`. Hasil proses ekstrak akan menghasilkan direktori ns-3-allinone. Isi dari direktori tersebut ditunjukkan pada Tabel V-9.

Tabel V-9. Direktori ns-3-allinone cara manual.

```
bake  build.py  constants.py  netanim-3.105  ns-3.21  pybindgen-
0.17.0.876  README  util.py
```

Pada Tabel V-9 dapat dilihat bahwa isi dari direktori ns-3-allinone hasil unduh dengan cara manual ternyata sama seperti pada Tabel V-7.

V.1.6 **Kompilasi Kode Sumber Ns-3**

Kompilasi kode sumber simulator ns-3 dapat dilakukan dengan dua cara. Kedua cara tersebut yaitu dengan menggunakan file script build.py atau dengan menggunakan perintah waf. Pada buku ini akan dibahas kedua cara tersebut. Kode sumber ns-3-allinone memiliki file build.py yang merupakan file konfigurasi default dari kode sumber ns-3 yang di dapatkan secara langsung ketika mengunduh kode sumber ns-3. Untuk cara manual, pengguna simulator ns-3 biasanya menggunakan native build system dalam konfigurasi yaitu dengan menggunakan perintah waf.

V.1.6.1 ***Kompilasi Menggunakan build.py***

Setelah proses unduh selesai, akan terdapat direktori ns-3-allinone di dalam direktori ~/repos. Jika pengguna simulator ns-3 mengunduh menggunakan tarball maka akan terdapat direktori ns-3-allinone-3.21 di dalam direktori ~/tarball. Perhatikan Tabel V-10.

Tabel V-10. Kompilasi menggunakan build.py.

```
ns-3@ubuntu:~/Public/repos/ns-3-allinone$ ./build.py --enable-
examples --enable-tests
```

Pada Tabel V-10, dengan memasukan argument `--enable-examples` dan `--enable-tests`, maka program-program contoh simulator ns-3 yang berada di dalam direktori examples ikut dikompilasi karena secara default simulator ns-3 tidak memasukan hal itu. Pengguna simulator ns-3 boleh untuk tidak menyertakan program-program contoh di direktori examples dan beberapa modul simulator ns-3. Pada Tabel

V-11 dapat dilihat hasil kompilasi yang dilakukan oleh script build.py. Informasi pada Tabel V-11 adalah pengguna dapat mengetahui waktu yang dibutuhkan untuk melakukan kompilasi, di mana dalam Tabel V-11 ditunjukkan proses kompilasi yang dibutuhkan sekitar 20 menit 55 detik. Terdapat modul yang secara default tidak disertakan dalam proses kompilasi diantaranya adalah modul `brite`, `click`, `opeflow` dan `visualizer`.

Tabel V-11. Hasil kompilasi kode sumber ns-3.

```
Waf: Leaving directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
'build' finished successfully (20m55.714s)
Modules built:
antenna                  aodv                       applications
bridge                   buildings                  config-store
core                     csma                       csma-layout
dsdv                     dsr                        energy
fd-net-device            flow-monitor               internet
lr-wpan                  lte                        mesh
mobility                 mpi                        netanim
network                  nix-vector-routing         olsr
point-to-point           point-to-point-layout      propagation
sixlowpan                spectrum                   stats
tap-bridge               test (no Python)           topology-read
uan                      virtual-net-device         wave
wifi                     wimax
Modules not built (see ns-3 tutorial for explanation):
brite                    click                      openflow
visualizer
```

V.1.6.2 ***Kompilasi Menggunakan waf***

Bagi pengguna simulator ns-3 menggunakan perintah waf merupakan hal yang mutlak diperlukan. Memang pengguna dapat menggunakan script, tetapi pada dasarnya script tersebut adalah perintah waf yang dibungkus ke dalam suatu file. Sebagai contoh, untuk dapat menggunakan modul examples dan test dengan perintah waf, dijalankan perintah seperti Tabel V-12.

Tabel V-12. Menggunakan perintah waf.

```
ns-3@ubuntu:~/Public/tarball/ns-allinone-3.21/ns-3.21$ ./waf clean
ns-3@ubuntu:~/Public/tarball/ns-allinone-3.21/ns-3.21$./waf -d
optimized --enable-examples --enable-tests configure
```

Perhatikan Tabel V-12, perintah pada baris pertama adalah suatu instruksi untuk menghapus file-file yang berada di direktori /build sebagai hasil kompilasi sebelumnya. Jika pengguna simulator ns-3 baru pertama mengunduh kode sumber ns-3 (fresh) hal ini

bisa saja dilewatkan dan perintah tersebut berguna untuk menghapus file objek dan perpustakaan sistem / library yang ada di dalam direktori /build. Ketika perintah ./waf dijalankan kompiler akan melakukan pendeteksian beberapa fitur-fitur yang dibutuhkan oleh kode sumber ns-3, seperti terlihat pada Tabel V-13.

Tabel V-13. Potongan kode konfigurasi ns-3 menggunakan waf.

```
Setting top to    : /home/ns-3/Public/tarball/ns-allinone-3.21/ns-3.21
Setting out to    : /home/ns-3/Public/tarball/ns-allinone-3.21/ns-
3.21/build
Checking for 'gcc' (c compiler)       : /usr/bin/gcc
Checking for cc version        : 4.6.3
Checking for 'g++' (c++ compiler)     : /usr/bin/g++
Checking for compilation flag -march=native... support          : ok
Checking for compilation flag -Wl,--soname=foo... support : ok
Checking for program python           : /usr/bin/python
..........................................
..........................................
---- Summary of optional NS-3 features:
Build profile                         : optimized
Threading Primitives                  : enabled
Real Time Simulator                   : enabled
Emulated Net Device                   : enabled
File descriptor NetDevice             : enabled
'configure' finished successfully (5.967s)
```

Pada Tabel V-13 dapat dilihat bahwa ada beberapa pilihan yang secara default tidak aktif dan membutuhkan dukungan perpustakaan sistem dari sistem operasi agar dapat berjalan dengan baik. Sebagai contoh untuk mengaktifkan `XmlTo` maka perpustakaan sistem / library `libxml-2.0` harus ada di sistem operasi. Hasil konfigurasi simulator ns-3 menggunakan perintah waf membutuhkan waktu sekitar 6 detik. Dan langkah selanjutnya adalah melakukan kompilasi kode sumber ns-3 dengan menjalankan perintah seperti pada Tabel V-14.

Tabel V-14. Kompilasi kode sumber menggunakan waf.

```
ns-3@ubuntu:~/Public/tarball/ns-allinone-3.21/ns-3.21$ ./waf
```

Setelah perintah ./waf dieksekusi, maka kompiler waf akan melakukan kompilasi semua modul yang berada di direktori ns-allinone-3.21. Hasil yang di peroleh setelah kompilasi yang dilakukan adalah sama seperti perintah build.py yaitu seperti pada Tabel V-15.

Tabel V-15. Hasil kompilasi kode sumber ns-3 menggunakan waf.

```
Waf: Leaving directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
'build' finished successfully (20m55.714s)
Modules built:
antenna                       aodv                       applications
bridge                    buildings                  config-store
core                      csma                       csma-layout
dsdv                          dsr                        energy
fd-net-device             flow-monitor               internet
lr-wpan                   lte                        mesh
mobility                  mpi                        netanim
network                   nix-vector-routing         olsr
point-to-point            point-to-point-layout      propagation
sixlowpan                 spectrum                   stats
tap-bridge                test (no Python)           topology-read
uan                       virtual-net-device         wave
wifi                      wimax
Modules not built (see ns-3 tutorial for explanation):
brite                     click                      openflow
visualizer
```

V.1.6.3 ***Testing Ns-3***

Pengguna simulator ns-3 dapat melakukan unit test dari distribusi ns-3 dengan menjalankan script ./test.py –core.

Tabel V-16. Unit Test simulator ns-3.

```
ns-3@ubuntu:~/Public/repos/ns-3-allinone/ns-3-dev$ ./test.py -core
```

Maksud dari menjalankan script ./test.py -core adalah untuk menguji fungsionalitas masing-masing modul yang telah berhasil di kompilasi. Pada Tabel V-17 dapat dilihat keluaran dari script test.py.

Tabel V-17. Hasil unit test simulator ns-3.

```
Waf: Entering directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
Waf: Leaving directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
'build' finished successfully (7.279s)
PASS: TestSuite attributes
PASS: TestSuite callback
PASS: TestSuite command-line
..........................
..........................
PASS: TestSuite timer
PASS: TestSuite traced-callback
PASS: TestSuite type-traits
```

Jika hasil unit test berhasil, maka akan tertulis kata PASS. Verifikasi simulator ns-3 oleh pengguna berguna untuk menyatakan bahwa distribusi simulator ns-3 yang terpasang berjalan dengan baik.

V.2 Menjalankan Script Simulasi pada Ns-3

Pada umumnya pengguna simulator ns-3 menjalankan script dengan menggunakan perintah waf. Untuk menjalankan script gunakan pilihan --run di waf. Sebagai contoh sederhana akan ditunjukan cara menjalankan script scratch-simulator menggunakan perintah waf. Jalankan perintah seperti ditunjukkan pada Tabel V-18, dan amati hasil menjalankan script tersebut pada Tabel V-19.

Tabel V-18. Menjalankan script scratch-simulator.cc pada simulator ns-3.

```
ns-3@ubuntu:~/Public/repos/ns-3-allinone/ns-3-dev$ ./waf --run
scratch/scratch-simulator
```

Tabel V-19. Hasil menjalankan script scratch-simulator.cc pada simulator ns-3.

```
Waf: Entering directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
Waf: Leaving directory `/home/ns-3/Public/repos/ns-3-allinone/ns-3-
dev/build'
'build' finished successfully (3.271s)
Scratch Simulator
```

Pada Tabel V-19, script scratch-simulator.cc pada simulator ns-3 telah berhasil dijalankan dengan baik. Hal ini berarti simulator ns-3 telah siap digunakan untuk mensimulasikan jaringan komputer.

V.3 Uji Coba Simulasi Jaringan

Simulator ns-3 dikembangkan dan didistribusikan secara lengkap menggunakan bahasa pemogramman C++ [28]. Untuk membuat program simulasi dalam simulator ns-3, pengguna harus menulis program dalam C++ atau dapat juga menggunakan python untuk membangun beberapa elemen-elemen yang mendeskripsikan komunikasi jaringan beserta aktivitas yang dibutuhkan oleh jaringan tersebut. Program dikompilasi dan

dilanjutkan dengan proses linking (mengaitkan) antara program dan perpustakaan sistem / library dari model-model jaringan yang ada di dalam simulator ns-3.

Dalam membuat program simulasi jaringan, terdapat empat langkah dasar yang harus dijalankan yaitu:

1. Membuat topologi jaringan yaitu membuat objek untuk node, devices, channel, dan protokol stack jaringan yang akan dimodelkan dalam simulasi.
2. Membuat data demand pada jaringan yaitu membuat model untuk beberapa aplikasi jaringan yang mengirim dan menerima data dari jaringan, yang menyebabkan paket tersebut diterima dan diproses.
3. Menjalankan simulasi yaitu simulator melakukan main event loop, yang menjalankan kejadian (event) yang telah dibentuk secara urut sebelumnya. Proses ini akan terus berjalan sampai daftar kejadian (event) kosong atau waktu berhenti telah sampai.
4. Menganalisa hasil yaitu melakukan analisa dari hasil trace setelah proses simulasi dijalankan. File-file trace berisi informasi yang dapat digunakan untuk melihat ip address, port, ukuran paket, menghitung link utilization pada komunikasi channel, rata-rata ukuran queue pada beberapa queue, dan rate di dalam queue.

Pada simulator ns-3, secara teknis terdapat beberapa struktur pemogramman yang harus dibuat sebagai berikut:

1. Boilerplate: keperluan dokumentasi
2. Module includes: menyertakan file-file header
3. Ns-3 namespace: deklarasi global
4. Logging: optional
5. Main function: deklarasi fungsi utama
6. Topology helpers: objects to combine distinct operations
7. Applications: on/off, UdpEchoClient/Server
8. Tracing: .tr and/or .pcap files
9. Simulator: start/end simulation, cleanup

Untuk mengilustrasikan langkah diatas dengan menggunakan simulator ns-3, sub-bab ini mengambil contoh script simulasi first.cc yang terdapat di direktori examples pada simulator ns-3. Pada script simulasi first.cc banyak menggunakan kelas container dan helper. Container adalah cara yang mudah untuk membuat, mengatur dan mengakses kelompok objek yang sama sedangkan helper membuat pekerjaan pemrograman dalam simulator ns-3 menjadi lebih mudah.

Tabel V-20. Script examples/tutorial/first.cc.

```
#include "ns-3/network-module.h"
#include "ns-3/internet-module.h"
#include "ns-3/point-to-point-module.h"
#include "ns-3/applications-module.h"
using namespace ns-3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

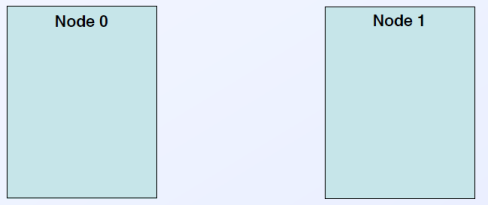
Pada Tabel V-20 dapat dlihat secara lengkap script first.cc. Script tersebut akan dibagi-bagi sesuai dengan fungsinya masing-masing. Hal ini dimaksudkan untuk memberikan gambaran secara jelas dari tiap-tiap kode. Fokus pertama adalah menjelaskan bagaimana cara membuat node pada simulator ns-3.

Tabel V-21. Pembuatan node pada simulator ns-3.

```
NodeContainer nodes;
nodes.Create(2);
```

Pada Tabel V-21, di deklarasikan objek `nodes` dengan tipe kelas `NodeContainer`. Hal ini berarti `nodes` memiliki sifat dan perilaku (properties dan method) dari kelas

tersebut. Objek `nodes` memanggil method `Create()` dengan memasukan parameter 2 untuk membuat 2 buah node. Ilustrasi pembuatan node dapat dilihat pada Gambar V-4.



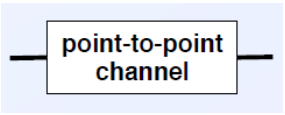
Gambar V-4. Ilustrasi pembuatan node.

Fokus kedua adalah menjelaskan bagaimana cara mendefinisikan channel point-to-point pada simulator ns-3.

Tabel V-22. Pembuatan channel pada simulator ns-3.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate",StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay",StringValue ("2ms"));
```

Pada Tabel V-22, di deklarasikan objek `pointToPoint` dengan tipe kelas `PointToPointHelper`. Hal ini berarti `pointToPoint` memiliki sifat dan perilaku (properties dan method) dari kelas tersebut. Objek `pointToPoint` memanggil method `SetDeviceAttribute()` dengan memasukan dua buah parameter string yaitu `"DataRate"` dan `"5Mbps"`. Maksud dari menggunakan method `SetDeviceAttribute()` adalah objek `pointToPoint` menentukan data rate channel point-to-point sebesar 5 Mbps dan kemudian dilanjutkan dengan memanggil method `SetChannelAttribute()` dengan memasukan dua buah parameter string yaitu `"Delay"` dan `"2ms"`. Maksud dari menggunakan method `SetDeviceAttribute()` adalah objek `pointToPoint` menentukan delay channel point-to-point sebesar 2 ms. Ilustrasi pembuatan channel dapat dilihat pada Gambar V-5.



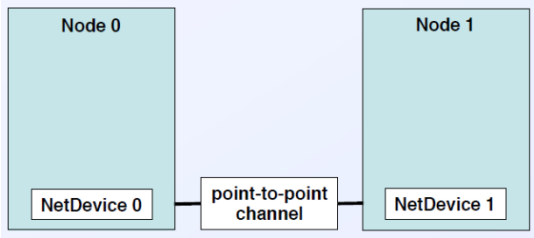
Gambar V-5. Channel point-to-point.

Fokus ketiga adalah menjelaskan bagaimana cara menghubungkan node, device jaringan dan channel point-to-point pada simulator ns-3.

Tabel V-23. Integrasi node, netdevice dan channel point-to-point.

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Pada Tabel V-23, dideklarasikan objek `devices` dengan tipe kelas `NetDeviceContainer`. Hal ini berarti `devices` memiliki sifat dan perilaku (properties dan method) dari kelas tersebut. Objek `devices` akan mewarisi setiap sifat dan perilaku dari objek `pointToPoint` karena objek `pointToPoint` memanggil method `Install()` dengan memasukan parameter objek `nodes` untuk mengintegrasikan node, channel dan device. Ilustrasi tersebut dapat dilihat pada Gambar V-6.



Gambar V-6. Integrasi node, netdevice dan channel point-to-point.

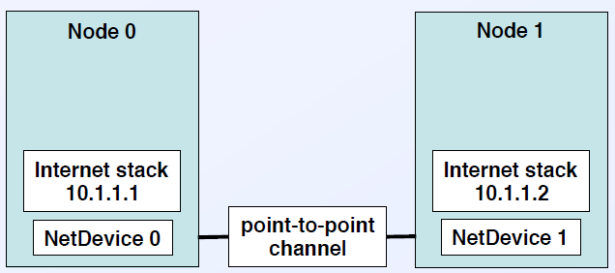
Fokus keempat adalah menjelaskan bagaimana cara menentukan alamat jaringan dan netmask dari jaringan yang akan disimulasikan pada simulator ns-3.

Tabel V-24. Menentukan alamat jaringan beserta netmask.

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Pada Tabel V-24, dideklarasikan objek `address` dengan tipe kelas `Ipv4AddressHelper`. Hal ini berarti `address` memiliki sifat dan perilaku (properties dan method) dari kelas tersebut. Objek `address` memanggil method `SetBase()` dengan memasukan parameter string `"10.1.1.0"` untuk alamat jaringan dan parameter string `"255.255.255.0"` untuk netmask. Method `SetBase()` akan secara otomatis mengorganisasikan ip address ke tiap-tiap interface. Kemudian dideklarasikan kembali objek `interfaces` dengan tipe kelas `Ipv4InterfaceContainer`. Hal ini berarti objek `interfaces` akan mewarisi sifat dan perilaku dari kelas `Ipv4InterfaceContainer`. Untuk dapat mengisi alamat ip ke tiap-tiap interface pada node maka objek `address`

memanggil method `Assign()` dengan parameter `devices` dan diwariskan ke objek `interfaces`. Ilustrasi tersebut dapat dilihat pada Gambar V-7.



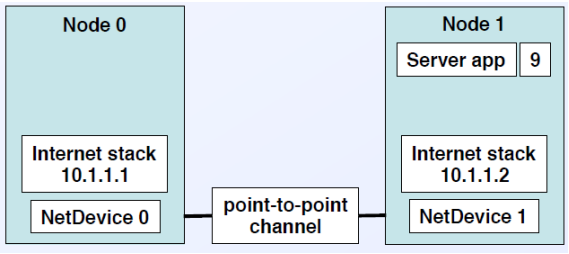
Gambar V-7. Menentukan alamat jaringan beserta netmask.

Fokus kelima adalah menjelaskan bagaimana cara membuat aplikasi di sisi server yang akan disimulasikan pada simulator ns-3.

Tabel V-25. Memasang aplikasi server pada node 1.

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
```

Pada Tabel V-25, dideklarasikan objek `echoServer` dan memasukan parameter 9 dengan tipe kelas `UdpEchoServerHelper`. Hal ini berarti `echoServer` memiliki sifat dan perilaku (properties dan method) dari kelas tersebut. Nilai 9 memiliki arti sebagai port yang di definisikan pada node. Untuk lebih jelasnya baca kembali pendefinisian port number di situs www.iana.org. Kemudian dideklarasikan kembali objek `serverApps` dengan tipe kelas `ApplicationContainer`. Hal ini berarti objek `serverApps` akan mewarisi sifat dan perilaku dari kelas `ApplicationContainer`. Untuk dapat memasang aplikasi server ke node maka objek `echoServer` memanggil method `Install()` dengan parameter `nodes` yang juga memanggil method `Get()` dengan parameter nilai 1 yang berarti node 1 dan kemudian diwariskan ke objek `serverApps`. Ilustrasi tersebut dapat dilihat pada Gambar V-8.



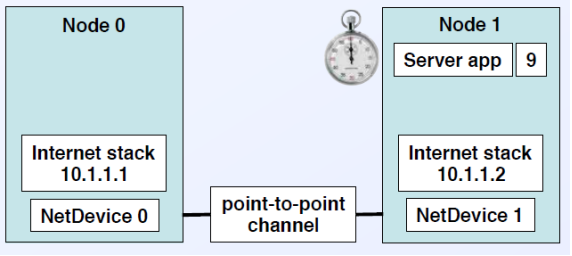
Gambar V-8. Memasang aplikasi server.

Fokus keenam adalah menjadwalkan aktifitas server aplikasi di sisi server yang akan disimulasikan pada simulator ns-3.

Tabel V-26. Penjadwalan aktifitas server.

```
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

Pada Tabel V-26, Objek `serverApps` memanggil method `Start()` dengan memasukan satu buah parameter nilai yaitu `Seconds (1.0)`. Maksud dari menggunakan method `Start()` adalah objek `serverApps` menentukan waktu dimulainya aktifitas server pada detik pertama 1s dan kemudian dilanjutkan dengan memanggil method `Stop()` dengan memasukan satu buah parameter nilai yaitu `Seconds (10.0)`. Maksud dari menggunakan method `Stop()` adalah objek `serverApps` menentukan waktu berhentinya aktivitas server pada detik kesepuluh 10s. Ilustrasi penjadwalan aktivitas server dapat dilihat pada Gambar V-9.



Gambar V-9. Penjadwalan aktivitas server.

Fokus ketujuh adalah menjelaskan bagaimana cara membuat aplikasi di sisi client yang akan disimulasikan pada simulator ns-3.

Tabel V-27. Memasang aplikasi client pada node 0.

```
UdpEchoClientHelper echoClient(interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
```

Pada Tabel V-27, dideklarasikan objek `echoClient` dengan tipe kelas `UdpEchoClientHelper`. Hal ini berarti `echoClient` memiliki sifat dan perilaku (properties dan method) dari kelas tersebut. Objek `echoClient` juga memasukan dua buah parameter yaitu `interfaces.GetAddress (1)` untuk menentukan ip address dan Nilai 9 untuk menentukan port. Kemudian objek `echoClient` memasukan dua buah parameter yaitu `"MaxPackets"` untuk mendefinisikan maksimum paket dan `UintegerValue (1)` sebagai jumlah paket yang dikirim, dalam hal ini paket yang dikirim adalah 1 buah. Objek `echoClient` memasukan parameter `"Interval"` untuk mendefinisikan interval atau batas waktu antar paket data yang dikirim, dalam hal ini ditentukan oleh parameter `TimeValue (Seconds (1.0))` dimana interval di setting 1 detik. Kemudian objek `echoClient` menentukan ukuran paket dengan memasukkan parameter `"PacketSize"` untuk mendefinisikan ukuran paket dan `UintegerValue (1024)` dimana ukuran paket data di setting 1024 Byte atau 1KByte. Untuk dapat memasang aplikasi client ke node maka objek `echoClient` memanggil method `Install()` dengan parameter `nodes` yang juga memanggil method `Get()` dengan parameter nilai 0 yang berarti node 0 dan kemudian diwariskan ke objek `clientApps`. Ilustrasi tersebut dapat dilihat pada Gambar V-10.

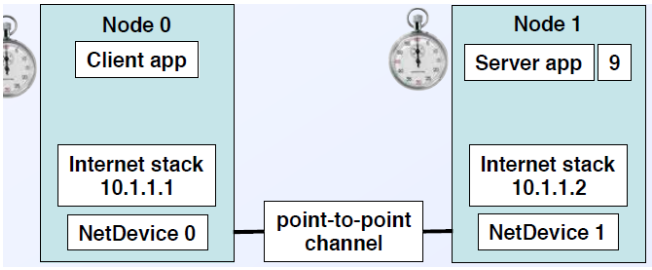
Fokus kedelapan adalah menjadwalkan aktivitas client aplikasi di sisi client yang akan disimulasikan pada simulator ns-3.

Tabel V-28. Penjadwalan aktivitas client.

```
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

Pada Tabel V-28, Objek `clientApps` memanggil method `Start()` dengan memasukan satu buah parameter nilai yaitu `Seconds (2.0)`. Maksud dari menggunakan method `Start()` adalah objek `clientApps` menentukan waktu dimulainya aktivitas client pada detik kedua 2s dan kemudian dilanjutkan dengan memanggil method `Stop()` dengan memasukan satu buah parameter nilai yaitu `Seconds (10.0)`. Maksud dari menggunakan method `Stop()` adalah objek

`clientApps` menentukan waktu berhentinya aktivitas client pada detik kesepuluh 10s. Ilustrasi penjadwalan aktivitas server dapat dilihat pada Gambar V-10.



Gambar V-10. Memasang aplikasi client.

Fokus kesembilan adalah menjalankan script simulasi dan menghapus objek jika proses simulasi telah selesai dijalankan pada simulator ns-3.

Tabel V-29. Menjalankan fungsi run() dan destroy() pada simulator ns-3.

```
Simulator::Run();
Simulator::Destroy ();
return 0;
```

Pada Tabel V-29, di definisikan fungsi-fungsi dari kelas `Simulator`. Yang pertama di definisikan fungsi `Run()` dan yang kedua fungsi `Destroy()`, kemudian program akan mengembalikan nilai 0 jika simulasi berjalan dengan baik. Script simulasi first.cc yang akan dijalankan terlebih dahulu di-copy ke direktori scratch kemudian dikompilasi dengan perintah ./waf. Jika kompilasi sukses dan tidak ada error, selanjutnya adalah menjalankan perintah ./waf --run scratch/myfirst dan untuk lebih jelasnya dapat dilihat pada Tabel V-30.

Tabel V-30. Menjalankan script myfirst.cc.

```
cp examples/tutorial/first.cc scratch/myfirst.cc
$ ./waf
$ ./waf –run scratch/myfirst
```

Output yang dihasilkan dari simulasi tersebut dapat dilihat kembali pada Tabel V-31.

Tabel V-31. Hasil simulasi myfirst.cc.

```
Waf: Entering directory `/mypath/ns-3-dev/build'
Waf: Leaving directory `/mypath/ns-3-dev/build'
'build' finished successfully (5.283s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```

Pada Tabel V-31, simulasi membutuhkan waktu 5.3 detik untuk dapat mensimulasikan script myfirst.cc.

Tabel V-32. Menambahkan proses tracing dengan format ASCII.

```
#include "ns-3/core-module.h"
#include "ns-3/network-module.h"
#include "ns-3/internet-module.h"
#include "ns-3/point-to-point-module.h"
#include "ns-3/applications-module.h"
using namespace ns-3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
/* tambahkan baris dibawah */
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll( ascii.CreateFileStream("myfirst.tr"));
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

Hal yang penting bagi simulasi adalah mengetahui informasi yang di-generate oleh simulator, dan simulator ns-3 telah dilengkapi dengan metode tracing. Tracing disediakan dalam output yang terstruktur dari simulator ns-3, dan dibangun berdasarkan prinsip source dan sink. Trace source adalah kejadian yang di-generate oleh simulator ns-3 dan menyediakan akses terhadap data, sedangkan sink adalah informasi dari proses trace. Terdapat dua tipe tracing yang disediakan oleh simulator ns-3 yaitu:

1. ASCII tracing : membuat file trace yang readable.
2. PCAP tracing : membuat file-file untuk tool analyzer (misal: tcpdump, wireshark).

Untuk dapat melakukan proses tracing, tambahkan beberapa kode program di dalam script myfirst.cc. Contoh kode program dari proses tracing dalam bentuk ASCII dapat dilihat pada Tabel V-32. Kemudian, jalankan perintah seperti ditunjukan pada Tabel V-33.

Tabel V-33. Menjalankan script myfirst.cc.

```
$ ./waf –run scratch/myfirst
```

Setelah proses simulasi dijalankan seperti pada Tabel V-33, secara otomatis simulator ns-3 akan membentuk file myfirst.tr di top level direktori dan masing-masing baris membentuk satu trace event. Tabel V-34 merupakan interpretasi dari baris pertama file myfirst.tr.

Tabel V-34. Hasil tracing pada file myfirst.tr.

```
+2 /NodeList/0/DeviceList/0/$ns-
3::PointToPointNetDevice/TxQueue/Enqueue
ns-3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns-
3::Ipv4Header
(tos 0x0 ttl 64 id 0 protocol 17 offset (bytes) 0 flags [none] length:
1052 10.1.1.1 > 10.1.1.2) ns-3::UdpHeader (length: 1032 49153 > 9)
Payload
(size=1024)
+ menyatakan kejadian (event)
2 menyatakan waktu simulasi dalam satuan detik (s)
/NodeList/0/DeviceList/0/$ns-3::PointToPointNetDevice/TxQueue/Enqueue
menyatakan sumber dari kejadian (event).
ns-3::PppHeader (Point-to-Point Protocol: IP (0x0021)) menyatakan
paket enkapsulasi paket.
ns-3::Ipv4Header (tos 0x0 ttl 64 id 0 protocol 17 offset (bytes) 0
flags [none] length: 1052 10.1.1.1 > 10.1.1.2) menyatakan info IP (IP
Adress)
ns-3::UdpHeader (length: 1032 49153 > 9) menyatakan info UDP (port)
Payload (size=1024) menyatakan data (payload)
```

Sedangkan untuk contoh kode program dari proses tracing dalam bentuk PCAP dapat dilihat pada Tabel V-35. Kemudian jalankan perintah seperti ditunjukan pada Tabel V-36.

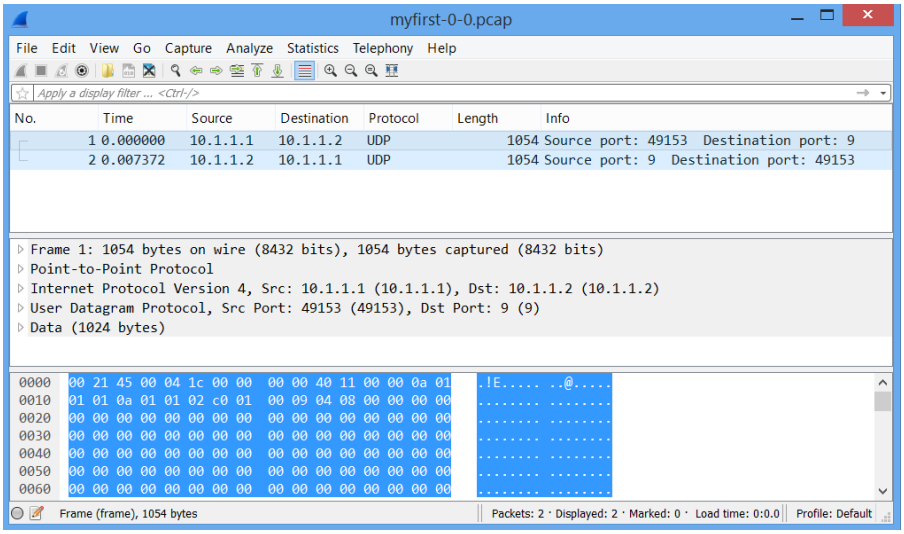
Tabel V-35. Menambahkan proses tracing dengan format PCAP.

```
#include "ns-3/core-module.h"
#include "ns-3/network-module.h"
#include "ns-3/internet-module.h"
#include "ns-3/point-to-point-module.h"
#include "ns-3/applications-module.h"
using namespace ns-3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
/* tambahkan baris dibawah */
pointToPoint.EnablePcapAll("myfirst");
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

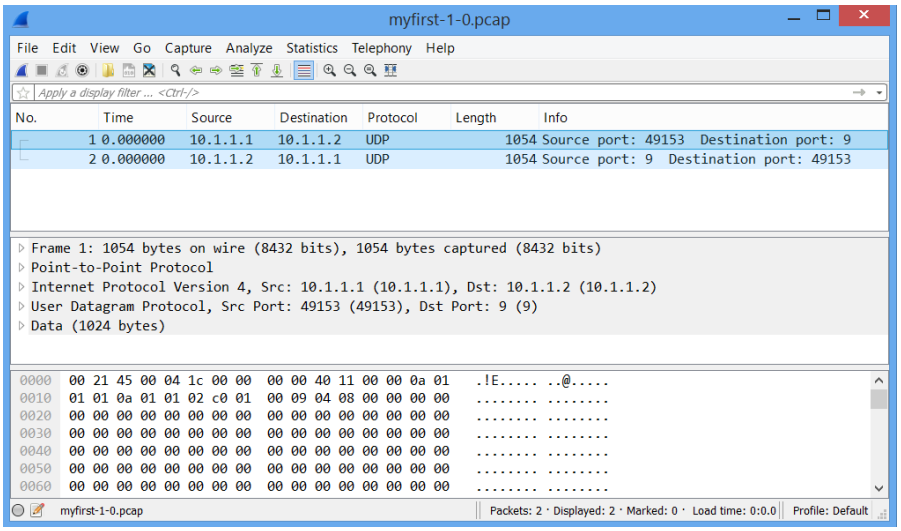
Tabel V-36. Menjalankan script myfirst.cc format PCAP.

```
$ ./waf –run scratch/myfirst
```

Perintah di atas akan membentuk file myfirst-0-0.pcap dan myfirst-0-1.pcap di top level direktori. Untuk dapat melihat hasil dari file tersebut dapat dilakukan dengan menggunakan perangkat lunak wireshark [34]. Coba perhatikan Gambar V-11 dan Gambar V-12.

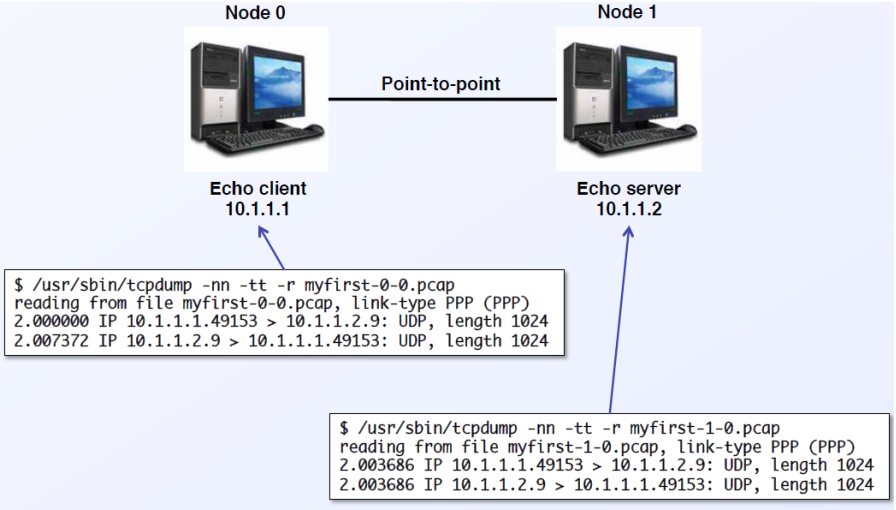


Gambar V-11. Analisa file myfirst-0-0.pcap menggunakan wireshark.



Gambar V-12. Analisa file myfirst-1-0.pcap menggunakan wireshark.

Isi dari file PCAP dapat juga dilihat menggunakan perangkat lunak tcpdump. Tcpdump adalah perangkat lunak network analyzer selain wireshark yang berbasis konsol dan sering digunakan untuk menganalisa paket-paket data. Hasil analisa file myfirst-x-x.pcap dengan menggunakan tcpdump dapat dilihat pada Gambar V-13.



Gambar V-13. Analisa file pcap menggunakan tcpdump.

Pada Gambar V-13, node 0 yang berfungsi sebagai client dengan ip address 10.1.1.1 dengan port 49153 mengirim packet data pada detik ke 2.000000s ke node 1 dengan ip address 10.1.1.2 port 9 dengan ukuran packet 1024 bytes atau 1KB menggunakan protokol user datagram protokol (UDP). Kemudian node 1 yang berfungsi sebagai server dengan ip address 10.1.1.2 port 9 merespons dengan mengirim packet balasan pada detik ke 2.007372 s ke node 0 dengan ip address 10.1.1.1 dan port 49153 dengan ukuran packet 1024 bytes atau 1KB menggunakan user datagram protokol (UDP). Jika melihat hasil capture packet pada file pcap menggunakan perangkat lunak tcpdump, kita dapat memperoleh informasi waktu, alamat pengirim dan penerima, port pengirim dan penerima, protokol yang digunakan beserta ukuran packet.

BAB VI

Pemogramman Berbasis Object Pada NS-3

VI.1 Pemogramman Berbasis Prosedural versus Objek

VI.1.1 Pemrograman Berbasis Prosedural

Bahasa pemrograman awalnya dirancang untuk berbasis prosedural, misal FORTRAN, COBOL, APL, PL/1, Ada, Basic, C, dan sebagainya. Programmer membuat sebuah program komputer dengan memerintahkan komputer berdasarkan pada baris perintah atau instruksi. Instruksi dibuat secara berurutan langkah demi langkah untuk membuat sebuah program komputer, dan programmer yang menentukan dan mengontrol urutan di mana komputer memproses instruksi. Dengan demikian, *urutan instruksi sangat penting*. Kelebihan dari bahasa pemogramman berbasis prosedural cenderung menghasilkan program yang berjalan dengan cepat dan pengelolaan sumber daya sistem yang digunakan secara efisien. Ini adalah pendekatan konvensional yang dipahami oleh banyak programmer, insinyur perangkat lunak, dan analis sistem [35].

VI.1.2 Pemogramman Berorientasi Objek

Pemogramman berorientasi objek (OOP) adalah sebuah pendekatan untuk merumuskan program sebagai rangkaian objek. Teknik pemogramman berorientasi objek memisahkan antara data dan metode yang terbungkus dalam objek untuk saling berinteraksi dalam melakukan tugas tertentu. Konsep pemrograman berorientasi objek sudah didukung oleh bahasa pemrograman modern seperti Java, C++, atau C#. Ini adalah konsep yang sama sekali berbeda dari pemrograman berbasis prosedural. Program berorientasi objek secara kognitif mirip dengan cara manusia yang memandang dunia nyata. Menggunakan teknik pemrograman berorientasi objek, pemrogram mungkin dapat memvisualisasikan solusi untuk suatu masalah menjadi lebih mudah. Aspek penting program berorientasi objek juga dapat meningkatkan efisiensi programmer karena teknologi enkapsulasi memungkinkan objek untuk diadopsi dan digunakan kembali dalam berbagai program yang berbeda. Kerugian dari program berorientasi objek adalah efisiensi runtime. Teknik pemograman berorientasi objek cenderung memerlukan lebih banyak memori dan pengolahan sumber daya dari pada teknik pemrograman berbasis prosedural.

VI.1.3 **Pemrograman Berorientasi Objek vs Prosedural**

Kembali ke bahasa berbasis prosedural seperti yang telah dijelaskan pada bagian VI.1.1, ada beberapa kelemahan dari teknik pemogramman berbasis prosedural yaitu pada cara memandang sebuah permasalahan. Pada era komputasi saat ini dimana perkembangan teknologi komputasi yang sangat kompleks dan beragam hampir menyelimuti segala aspek kehidupan masyarakat sangat dirasakan bahwa permasalahan yang timbul saat ini dirasakan kurang cocok diselesaikan dengan menggunakan teknik pemogramman berbasis prosedural. Beberapa jenis masalah yang timbul seperti memaksa programmer untuk melihat masalah sebagai serangkaian langkah. Namun, pada kenyataanya beberapa masalah mungkin lebih baik divisualisasikan sebagai objek.

Bahasa berorientasi objek yang telah ada sejak tahun 1960-an, pada 10 tahun terakhir menunjukkan pertumbuhan yang sangat signifikan dalam penggunaan teknologi berorientasi objek di seluruh industri perangkat lunak. Banyak programmer dan pengusaha industri perangkat lunak sekarang telah menganut konsep pemrograman berorientasi objek. Teknik pemrograman prosedural telah ditinggalkan dan mereka lebih memilih pemrograman berorientasi objek dalam mengembangkan sistem informasi. Namun, ada beberapa keuntungan dan kerugian dalam pemrograman berorientasi objek dengan teknik pemrograman berbasis prosedural yang tidak bisa diabaikan, secara umum teknik pemrograman berorientasi objek mengarah ke pengembangan program yang lebih cepat (efisiensi pengembangan), sedangkan teknik pemrograman berbasis prosedural mengarah kepada kecepatan program (efisiensi runtime) [35].

Dalam pemrograman berbasis prosedural, kode program dibuat secara berurutan (*sequential*). Sebagian besar, program berbasis prosedural memiliki variabel dan fungsi yang bersifat global (seperti di C) walaupun fungsi sendiri memiliki variabel lokal. Ini sangat berbeda dengan pemogramman berorientasi objek. Dalam pemograman berorientasi objek, program dibuat berdasarkan 'Objects'. Setiap objek memiliki metode sendiri (fungsi) dan variabel. Kebanyakan variabelnya tidak global.

VI.2 Gambaran Pemrograman Berorientasi Objek (OOP)

Pemrograman berorientasi objek bukanlah bahasa pemrograman melainkan sebuah cara untuk menjadikan program yang kita program menjadi lebih modular karena suatu permasalahan akan dikumpulkan dalam satu objek, yang selanjutnya akan disebut dengan kelas (class) [36]. Semua bahasa pemogramman yang mendukung OOP haruslah memiliki kemampuan untuk melakukan abstraksi, pembungkusan, pewarisan

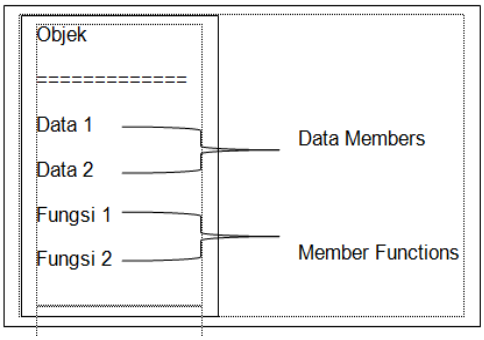
sifat dan poliforfisme. Karena sebagai ciri-ciri bahasa yang mendukung OOP. Prinsip-prinsip dasar di balik penggunaan pemrograman berorientasi objek adalah sebagai berikut.

VI.2.1 **Abstraksi (Abstraction)**

Abstraksi merupakan proses penyembunyian detil program yang sangat rumit sehingga kita tidak perlu mengetahui pembuatannya [36]. Yang dibutuhkan adalah pengetahuan terhadap objek yang memiliki fungsi tersebut dan cara penggunaannya saja. Sebagai contoh, misalnya objek mobil. Pembuat mobil tidak perlu mendefinisikan bagaimana cara pembuatannya, di sini berarti dia telah melakukan proses abstraksi yang terdapat di dalamnya. Yang penting bagi kita adalah mobil tersebut dapat kita gunakan sebagaimana mestinya. Hal ini tidaklah berbeda dengan proses abstraksi suatu fungsi yang terdapat pada sebuah kelas di dalam program.

VI.2.2 **Pembungkusan (Encapsulation)**

Pembungkusan adalah sebuah proses penggabungan antara data-data dan fungsi-fungsi yang berkaitan ke dalam sebuah objek [36]. Dengan demikian kita dapat membuat program yang terintegrasi, tanpa harus melakukan deklarasi variabel-variabel yang bersifat eksternal. Adapun istilah yang digunakan untuk menyebut data-data yang terdapat dalam suatu objek adalah *data members*, sedangkan fungsi-fungsi yang terdapat di dalamnya dikenal dengan istilah *member function* [36]. Untuk lebih memahami perhatikan ilustrasi yang ditunjukkan pada Gambar VI-1.



Gambar VI-1. Ilustrasi proses pembungkusan (encapsulation).

VI.2.3 **Pewarisan (Inheritance)**

Dalam pemrograman berorientasi objek, terdapat metode pembuatan objek baru yang diturunkan dari objek lain. Objek baru ini sering disebut dengan objek turunan (derived

class) sedangkan objek barunya sering disebut dengan ancestor (base class). Sifat yang terkandung dalam objek turunan adalah sifat yang diwariskan dari sifat yang dimiliki oleh objek induk. Maka dari itu proses tersebut sering dikenal dengan istilah pewarisan (*inheritance*). Dengan fitur ini, kita dapat membuat objek baru yang memiliki kemampuan lebih dibanding objek induknya, yaitu dengan cara menambahkan sifat-sifat baru ke dalam objek tersebut [36].

VI.2.4 **Polimorfisme (Polymorphism)**

Poliformisme adalah proses banyak rupa, artinya kita dapat mengimplementasikan sesuatu hal yang berbeda dengan cara yang sama [36]. Untuk mengerti hal ini, suatu objek, misalnya penyanyi. Sebagai contoh terdapat 5 orang penyanyi (dalam program dianggap sebagai 5 buah objek), kemudian kelimanya kita perintahkan untuk bernyanyi, maka hasil atau implementasinya tentu akan beda sesuai dengan karakteristik suara dari masing-masing penyanyi tersebut. Dalam pemogramman berorientasi objek, hal ini disebut dengan polimorfisme.

VI.3 Kelas dan Objek Pada NS-3

Kelas dibuat untuk merepresentasikan sebuah objek tertentu sehingga akan membantu dalam proses penyelesaian masalah-masalah kompleks. Apabila kita amati, pendefinisian kelas itu sama saja dengan pendefinisian sebuah tipe data baru. Kelas merupakan bentuk penyederhanaan dari suatu permasalahan yang berkaitan dengan objek. Maka dari itu kelas dapat didefinisikan sebagai sesuatu yang mempunyai data (sifat) dan fungsi (kelakuan). Kelas masih bersifat abstrak, oleh karena itu kita harus melakukan instansiasi dari kelas tersebut, selanjutnya instance (contoh nyata atau perwujudan) dari kelas tersebut juga sering dinyatakan dengan objek [36]. Sebagai contoh manusia adalah suatu kelas, maka instance atau objek dari kelas manusia adalah Bima, Kristian, Mery dan lain sebagainya. Pada simulator ns-3 yang menggunakan bahasa pemogramman C++, kelas dibuat dengan menggunakan kata kunci **class**. Adapun bentuk umum pembuatannya adalah seperti pada Tabel VI-1.

Tabel VI-1. Pembuatan kelas pada c++

```
class nama_kelas {
   accses_specifier1;
      data_members;
      member_functions;
      ...
   access_specifier2;
      data_members;
      member_functions;
      ...
   ...
};
```

Untuk mendefinisikan atau membuat implementasi fungsi-fungsi yang terdapat dalam sebuah kelas, digunakan operator **::** dan berikut ini adalah bentuk umum dari pendefinisian fungsi. Lihat Tabel VI-2.

Tabel VI-2. Pendefinisian fungsi pada c++

```
tipe data nama_kelas::nama_fungsi(daftar parameter) {
   Statement yang akan dilakukan;
   ...
}
```

Sedangkan untuk mengakses data atau fungsi yang terdapat di dalam kelas tersebut, dapat menggunakan tanda titik **.** dan berikut ini bentuk umum dari proses pengaksesan data atau fungsi dari sebuah kelas. Contoh ditunjukkan pada Tabel VI-3.

Tabel VI-3. Mengakes data atau fungsi pada c++

```
nama_instance.data
nama_instance.nama_fungsi(daftar_parameter)
```

Lihat kembali pada bagian V.3 tentang uji coba simulasi, di mana buku ini menggunakan contoh kelas dan objek yang digunakan dalam script myfirst.cc. Lihat kembali bagaimana objek node dibentuk pada Tabel VI-4.

Tabel VI-4. Deklarasi node.

```
NodeContainer nodes;
```

Pada Tabel VI-4, di deklarasikan variabel `nodes` dengan tipe kelas `NodeContainer`. Lihat lebih detil pada Tabel VI-5 member data dan member fungsi kelas `NodeContainer`.

Tabel VI-5. Potongan kelas NodeContainer.

```
class NodeContainer
{
public:
  NodeContainer ();
  NodeContainer (Ptr<Node> node);
  NodeContainer (std::string nodeName);
  ...
  ...
  void Create (uint32_t n);
  ...
private:
  std::vector<Ptr<Node> > m_nodes;
};
```

Pada Tabel VI-5, potongan kode kelas `NodeContainer` memiliki 4 method dan 1 properties. Definisi detil dari ke empat method tersebut disembunyikan dan hal ini sesuai dengan prinsip abstraksi dan pembungkusan pada pemogramman berorientasi objek. Jika ingin melihat detil implementasi dari beberapa method tersebut dapat dilihat pada file `node-container.cc`. Pada baris terakhir terdapat deklarasi data atau properties dari kelas `NodeContainer` yaitu `std::vector<Ptr<Node> > m_nodes`.

Objek `nodes` yang terbentuk dari kelas `NodeContainer` akan memiliki semua data (sifat) dan method (kelakuan) dari kelas tersebut. Coba perhatikan Tabel VI-6.

Tabel VI-6. Penggunaan method Create oleh objek nodes.

```
nodes.Create(2);
```

Pada Tabel VI-6 objek `nodes` memanggil method `Create` dengan memasukan argumen nilai 2 untuk membuat 2 buah node. Method `Create` merupakan member function dari kelas `NodeContainer` dan detil dari definisi method `Create` dapat dilihat pada Tabel VI-7.

Tabel VI-7. Pembuatan objek node menggunakan method Create.

```
void
NodeContainer::Create (uint32_t n)
{
  for (uint32_t i = 0; i < n; i++)
    {
      m_nodes.push_back (CreateObject<Node> ());
    }
}
```

Objek `nodes` memiliki semua karakteristik dari kelas `NodeContainer` karena pada simulator ns-3 telah memenuhi prinsip-prinsip pemogramman berorientasi objek yaitu pembungkusan (encapsulation). Lihat kembali bagaimana jaringan point-to-point dibentuk.

Tabel VI-8. Deklarasi PointToPoint.

```
PointToPointHelper pointToPoint;
```

Pada Tabel VI-8 dideklarasikan variabel pointToPoint dengan tipe kelas `PointToPointHelper`. Lihat lebih detil pada Tabel VI-9 member data dan member fungsi kelas `PointToPointHelper`.

Tabel VI-9. Potongan kode kelas PointToPointHelper.

```
class PointToPointHelper : public PcapHelperForDevice, public
AsciiTraceHelperForDevice
{
public:
  PointToPointHelper ();
  virtual ~PointToPointHelper () {}
  ...
  ...
  void SetDeviceAttribute (std::string name, const AttributeValue
&value);
  void SetChannelAttribute (std::string name, const AttributeValue
&value);
private:
  ...
  ObjectFactory m_channelFactory;
};
```

Pada Tabel VI-9, potongan kode kelas `PointToPointHelper` memiliki 1 construktor dan destruktor, 2 method dan 1 properties. Definisi detil dari konstruktor dan destruktor beserta kedua method tersebut disembunyikan dan hal ini sesuai dengan prinsip-prinsip pemrograman berorientasi objek yaitu abstraksi (abstraction). Coba perhatikan pada

definisi kelas PointToPointHelper pada file `point-to-point-helper.cc` seperti pada Tabel VI-10.

Tabel VI-10. Definisi kelas PointToPointHelper.

```
class PointToPointHelper : public PcapHelperForDevice, public
AsciiTraceHelperForDevice
```

Pada Tabel VI-10, ternyata kelas `PointToPointHelper` pada simulator ns-3 merupakan turunan (derived class) dari kelas induk (base class) `PcapHelperForDevice` dan `AsciiTraceHelperForDevice`. Hal ini sesuai dengan prinsip-prinsip pemrograman berorientasi objek yaitu pewarisan (inheritance). Jika ingin melihat detil implementasi dari beberapa method tersebut dapat dilihat pada file `point-to-point-helper.cc`. Pada baris terakhir terdapat deklarasi data atau properties dari kelas `PointToPointHelper` yaitu `ObjectFactory m_channelFactory`.

Objek `pointToPoint` yang terbentuk dari kelas `PointToPointHelper` akan memiliki semua data (sifat) dan method (kelakuan) dari kelas tersebut. Coba perhatikan kode pada Tabel VI-11.

Tabel VI-11. Mendefinisikan data rate.

```
pointToPoint.SetDeviceAttribute ("DataRate",StringValue ("5Mbps"));
```

Pada Tabel VI-11, objek `pointToPoint` memanggil method `SetDeviceAttribute` dengan memasukan dua argumen. Method ini berfungsi untuk menentukan data rate dari jaringan point to point sebesar 5 Mbps di dalam simulasi. Method `SetDeviceAttribute` merupakan member function dari kelas `PointToPointHelper` dan detil dari definisi method `SetDeviceAttribute` dapat dilihat pada Tabel VI-12.

Tabel VI-12. Pendefinisian data rate menggunakan method SetDeviceAttribute.

```
void
PointToPointHelper::SetDeviceAttribute (std::string n1, const
AttributeValue &v1)
{
  m_deviceFactory.Set (n1, v1);
}
```

Coba perhatikan Tabel VI-13 yang mendefinisikan delay dari channel point-to-point yang telah didefinisikan sebelumnya.

Tabel VI-13. Mendefinisikan delay.

```
pointToPoint.SetChannelAttribute ("Delay",StringValue ("2ms"));
```

Pada Tabel VI-13, objek `pointToPoint` memanggil method `SetChannelAttribute` dengan memasukan dua argumen. Method `SetChannelAttribute` berfungsi untuk menentukan delay dari jaringan point to point sebesar 2 ms di dalam simulasi. Method `SetChannelAttribute` merupakan member function dari kelas `PointToPointHelper` dan detil dari definisi method `SetChannelAttribute` dapat dilihat pada Tabel VI-14.

Tabel VI-14. Pendefinisian delay menggunakan method SetChannelAttribute.

```
Void
PointToPointHelper::SetChannelAttribute (std::string n1, const
AttributeValue &v1)
{
  m_channelFactory.Set (n1, v1);
  m_remoteChannelFactory.Set (n1, v1);
}
```

Coba perhatikan kelanjutan kode script myfirst.cc dibawah ini tentang bagaimana objek devices yang mewakili device jaringan dalam simulator ns-3 dibentuk. Lihat Tabel VI-15.

Tabel VI-15. Deklarasi objek devices.

```
NetDeviceContainer devices;
```

Pada Tabel VI-15, dideklarasikan variabel `devices` dengan tipe kelas `NetDeviceContainer`. Lihat lebih detil pada Tabel VI-16 member data dan member fungsi kelas `NetDeviceContainer`.

Tabel VI-16. Potongan kode kelas NetDeviceContainer.

```
class NetDeviceContainer
{
public:
  NetDeviceContainer ();
  NetDeviceContainer (Ptr<NetDevice> dev);
  NetDeviceContainer (std::string devName);
  ...
  ...
private:
   std::vector<Ptr<NetDevice> > m_devices;
};
```

Pada Tabel VI-16, potongan kode kelas `NetDeviceContainer` memiliki 3 method dan 1 properties. Definisi detil dari ketiga method tersebut disembunyikan. Jika ingin melihat definisi detil dari beberapa method tersebut dapat dilihat pada file `net-device-container.cc`. Pada baris terakhir terdapat deklarasi data atau properties dari kelas `NetDeviceContainer` yaitu `std::vector<Ptr<NetDevice> > m_devices`.

Objek `devices` yang telah terbentuk dari kelas `NetDeviceContainer` akan memiliki semua data (sifat) dan method (kelakuan) dari kelas tersebut. Coba perhatikan kode dibawah ini. Lihat Tabel VI-17.

Tabel VI-17. Memasang channel point-to-point ke device pada tiap-tiap node.

```
devices = pointToPoint.Install (nodes);
```

Pada Tabel VI-17, objek `pointToPoint` memanggil method `Install` dengan memasukan argumen objek `nodes`. Method `Install` berfungsi untuk memasang device jaringan PPP ke dalam node di dalam simulasi. Kemudian objek `pointToPoint` mewariskan karakteristiknya pada objek `devices`. Method `Install` merupakan member function dari kelas `PointToPointHelper` dan detil dari definisi method `Install` adalah seperti ditunjukkan pada Tabel VI-18.

Tabel VI-18. Definisi fungsi install.

```
PointToPointHelper::Install (NodeContainer c)
{
  NS_ASSERT (c.GetN () == 2);
  return Install (c.Get (0), c.Get (1));
}
```

Kelanjutan kode pada script myfirst.cc adalah menentukan alamat ip (ip address) pada masing-masing node. Coba perhatikan kode dibawah ini.

Tabel VI-19. Deklarasi variabel address.

```
Ipv4AddressHelper address;
```

Pada Tabel VI-19, dideklarasikan variabel address dengan tipe kelas `Ipv4AddressHelper`. Lihat lebih detil pada kelas `Ipv4AddressHelper` untuk melihat beberapa pendefinisian dari kelas tersebut pada Tabel VI-20.

Tabel VI-20. Definisi kelas Ipv4AddressHelper.

```
class Ipv4AddressHelper
{
public:
  Ipv4AddressHelper ();
  Ipv4AddressHelper (Ipv4Address network, Ipv4Mask mask, Ipv4Address
base = "0.0.0.1");
  ...
  void SetBase (Ipv4Address network, Ipv4Mask mask, Ipv4Address base =
"0.0.0.1");
private:
  uint32_t m_network;
  uint32_t m_mask;
  uint32_t m_address;
  uint32_t m_base;
  ...
};
```

Pada Tabel VI-20, potongan kode kelas `Ipv4AddressHelper` memiliki 1 construktor, 2 method dan 4 properties. Implementasi detil dari konstruktor beserta kedua method tersebut disembunyikan dan hal ini sesuai dengan prinsip-prinsip pemrograman berorientasi objek yaitu abstraksi (abstraction). Objek `address` yang telah terbentuk dari kelas `Ipv4AddressHelper` akan memiliki semua data (sifat) dan method (kelakuan) dari kelas tersebut. Coba perhatikan kode dibawah ini.

Tabel VI-21. Menentukan alamat ip otomatis.

```
address.SetBase ("10.1.1.0", "255.255.255.0");
```

Pada Tabel VI-21, objek `address` memanggil method `SetBase` untuk menentukan alamat ip (ip address) 10.1.1.0 dengan subnetmask 255.255.255.0 di dalam simulasi. Method `SetBase` merupakan member function dari kelas `Ipv4AddressHelper`. Kelas `Ipv4AddressHelper` mengalokasikan alamat ip (ip address) secara otomatis

berdasarkan angka yang diberikan ke jaringan. Sebagai contoh, jika kita menginginkan pemberian alamat ip 192.168.1.1 dengan prefix /24 dan menginginkan alamat ip address pertama dimulai dengan ip 192.168.1.3 maka penggunaan method `SetBase` adalah seperti pada Tabel VI-22.

Tabel VI-22. Menentukan alamat ip secara manual.

```
address.SetBase ("192.168.1.0", "255.255.255.0", "0.0.0.3");
```

Secara default base address pada kelas `Ipv4AddressHelper` adalah "0.0.0.1". Lanjut ke kode berikutnya dari script myfirst.cc.

Tabel VI-23. Mendefinisikan interfaces.

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Pada Tabel VI-23, objek interfaces dibuat dari kelas Ipv4InterfaceContainer. Lihat lebih detil pendefinisian dari kelas Ipv4InterfaceContainer pada Tabel VI-24.

Tabel VI-24. Definisi kelas Ipv4InterfaceContainer.

```
class Ipv4InterfaceContainer
{
public:
  ...
  Ipv4InterfaceContainer ();
  void Add (Ipv4InterfaceContainer other);
  ...
private:
  typedef std::vector<std::pair<Ptr<Ipv4>,uint32_t> > InterfaceVector;
  InterfaceVector m_interfaces;
};
```

Pada Tabel VI-24, potongan kode kelas Ipv4InterfaceContainer memiliki 2 method dan 2 properties. Implementasi detil dari kedua method tersebut disembunyikan. Jika ingin melihat detil implementasi dari beberapa method tersebut dapat dilihat pada file ipv4-interface-container.cc. Pada baris terakhir terdapat deklarasi dari dua member data atau memiliki dua properties yaitu typedef std::vector<std::pair<Ptr<Ipv4>,uint32_t> > InterfaceVector dan InterfaceVector m_interfaces.

VI.4 Pewarisan Objek pada Ns-3 (Object Inheritance pada Ns3)

Salah satu ciri pemogramman berorientasi objek adalah pada objek atau kelas yang memiliki kemampuan untuk mewariskan sifat-sifat yang terdapat di dalamnya kepada kelas turunannya [36]. Hal yang paling penting dalam pewarisan adalah pemberian hak akses terhadap turunannya. Simulator ns-3 sudah mendukung konsep pewarisan.

VI.4.1 Kelas Dasar dan Kelas Turunan Pada Ns-3

Kelas dasar (base class) pada simulator ns-3 adalah sebuah kelas yang akan dijadikan sebagai induk dari kelas lain. Sedangkan kelas baru yang merupakan hasil dari proses penurunan tersebut disebut kelas turunan (derived class). Coba amati kelas Node dan NetDevice pada Tabel VI-25

Tabel VI-25. Kelas turunan Node.

```
class Node : Public Object
```

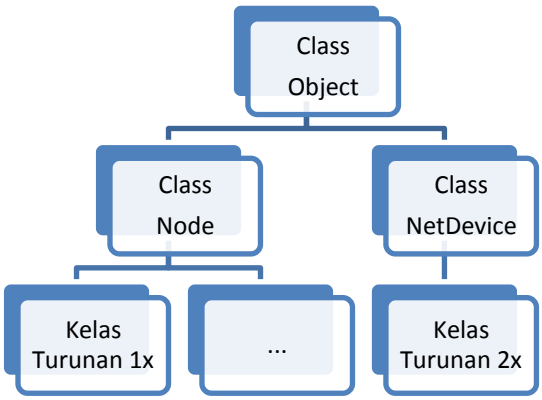
Kode pada Tabel VI-25 menunjukkan bahwa kelas Node merupakan kelas turunan dari kelas Object.

Tabel VI-26. Kelas turunan NetDevice.

```
class NetDevice : Public Object
```

Kode pada Tabel VI-26 menunjukkan pula bahwa kelas NetDevice merupakan kelas turunan dari kelas objek. Hal ini sesuai dengan prinsip-prinsip pemogramman berorientasi objek yaitu pewarisan (inheritance).

Penurunan sebuah kelas dalam simulator ns-3 ditunjukkan pada Gambar VI-2, di mana terlihat bahwa sebuah kelas dasar pada ns-3 dapat diturunkan menjadi beberapa kelas yang berbeda. Selanjutnya beberapa kelas tersebut dapat diturunkan lagi menjadi kelas-kelas turunan berikutnya, dan seterusnya.



Gambar VI-2. Pewarisan kelas pada simulator ns-3.

VI.4.2 **Hak Akses Pada Proses Pewarisan di Ns-3**

Pada simulator ns-3, proses pewarisan suatu kelas ditandai dengan memberikan hak akses data atau fungsi terhadap kelas turunan. Berikut ini hal-hal yang perlu diketahui dalam membuat sebuah kelas turunan. Apabila kelas diturunkan sebagai **public** dari kelas induknya, maka:

1. Bagian public yang terdapat pada kelas induk tetap akan menjadi bagian public pada kelas turunannya.
2. Bagian protected yang terdapat pada kelas induk tetap akan menjadi bagian protected pada kelas turunannya.
3. Bagian private yang terdapat pada kelas induk tetap akan menjadi bagian private pada kelas turunannya.

Apabila kelas diturunkan sebagai **private** dari kelas induknya, maka:

1. Bagian public yang terdapat pada kelas induk akan menjadi bagian private pada kelas turunannya.
2. Bagian protected yang terdapat pada kelas induk akan menjadi bagian private pada kelas turunannya.
3. Bagian private yang terdapat pada kelas induk tetap tidak akan dapat diakses oleh kelas turunannya.

Coba amati kembali script myfirst.cc, terdapat statement untuk membentuk channel point-to-point dan stack TCP/IP seperti pada Tabel VI-27.

Tabel VI-27. Objek pointToPoint dengan tipe kelas PointToPointHelper.

```
PointToPointHelper pointToPoint;
```

Pada Tabel VI-27, objek pointToPoint dibuat dari kelas PointToPointHelper. Lihat lebih detil pada kelas PointToPointHelper untuk melihat beberapa pendefinisian dari kelas PointToPointHelper.

Tabel VI-28. Kelas turunan PointToPointHelper.

```
class PointToPointHelper :     public PcapHelperForDevice, public
AsciiTraceHelperForDevice
```

Kelas PointToPointHelper pada simulator ns-3 merupakan turunan (derived class) dari kelas induk (base class) PcapHelperForDevice dan AsciiTraceHelperForDevice. Dengan menggunakan keyword **public,** maka bagian public pada kelas PcapHelperForDevice dan AsciiTraceHelperForDevice akan menjadi bagian public pada kelas PointToPointHelper. Begitupun pada pada bagian protected dan private kelas turunan akan mewarisi bagian protected dan private dari kelas induknya. Selanjutnya coba amati kode dibawah ini.

Tabel VI-29. Objek stack dengan tipe kelas InternetStackHelper.

```
InternetStackHelper stack;
```

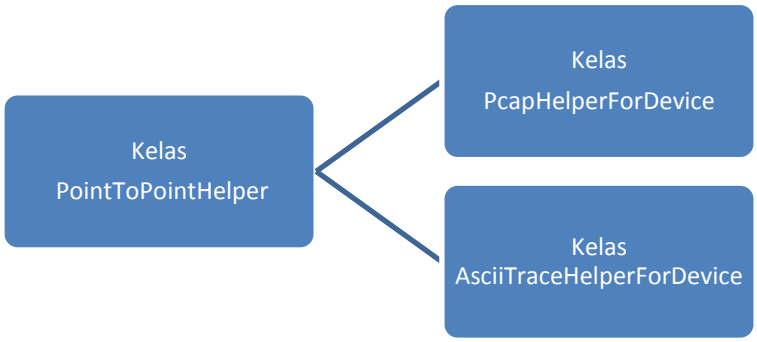
Pada Tabel VI-29 objek stack dibuat dari kelas InternetStackHelper. Lihat lebih detil pada kelas InternetStackHelper untuk melihat beberapa pendefinisian dari kelas InternetStackHelper.

Tabel VI-30. Kelas turunan InternetStackHelper.

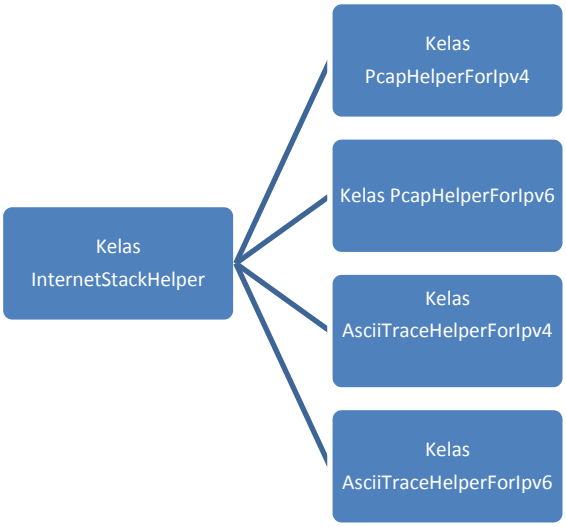
```
class InternetStackHelper : public PcapHelperForIpv4, public
PcapHelperForIpv6, public AsciiTraceHelperForIpv4, public
AsciiTraceHelperForIpv6
```

Kelas InternetStackHelper pada simulator ns-3 merupakan turunan (derived class) dari kelas induk (base class) PcapHelperForIpv4, PcapHelperForIpv6, AsciiTraceHelperForIpv4, dan AsciiTraceHelperForIpv6. Dengan menggunakan keyword public, maka bagian public pada keempat kelas induk akan menjadi bagian public pada kelas InternetStackHelper. Begitupun pada bagian protected dan private kelas turunan

akan mewarisi bagian protected dan private dari kelas induknya. Dalam hal ini kelas InternetStackHelper banyak mewarisi data (sifat) dan fungsi (perilaku) dari keempat kelas induk diatas.



Gambar VI-3. Pewarisan Ganda Kelas PointToPointHelper.



Gambar VI-4. Pewarisan Ganda Kelas InternetStackHelper.

VI.4.3 **Pewarisan Ganda Pada Ns-3 (Multiple Inheritance)**

Simulator ns-3 yang dibuat menggunakan bahasa C++ memiliki beberapa kelebihan. Salah satu diantaranya adalah karena C++ mendukung adanya multiple inheritance. Adapun multiple inheritance yang dimaksud disini adalah suatu proses pembuatan kelas baru dimana kelas tersebut diturunkan dari dua kelas induk atau lebih. Sebagai contoh, kita memiliki kelas A dan B yang masing-masing berdiri sendiri. Kemudian kita ingin membuat kelas C yang merupakan turunan dari kelas A dan B. Dengan demikian kelas C

akan mewarisi sifat-sifat yang terdapat pada kelas A dan B. Coba amati kembali sub-bab 6.4.2 tentang pewarisan di ns-3. Pada beberapa contoh kode di sub-bab tersebut pembuatan kelas PointToPointHelper dan InternetStackHelper merupakan kelas baru yang mewarisi dua atau lebih dari kelas induknya (multiple inheritance). Gambar VI-3 dan Gambar VI-4 mengilustrasikan proses multiple inheritance pada simulator ns-3.

VI.5 Fungsi Virtual dan Polimorfisme (Virtual dan Polymorphism)

Pada pemogramman berorientasi objek terdapat konsep yang disebut dengan polimorfisme. Namun sebelum itu akan dibahas mengenai hal mutlak yang diperlukan dalam mempelajari polimorfisme terlebih dahulu, yaitu fungsi virtual.

VI.5.1 **Fungsi Virtual**

Fungsi virtual adalah fungsi yang mendukung adanya *polymorphic function,* artinya fungsi tersebut dapat didefinisikan ulang pada kelas-kelas turunannya. Dalam C++, untuk mendefinisikan fungsi sebagai fungsi virtual adalah dengan menggunakan kata kunci **virtual**, yaitu dengan menempatkannya di depan pendeklarasian fungsi tersebut. Pendefinisian fungsi virtual yang terdapat pada kelas dasar biasanya tidak begitu berarti, artinya kode-kode yang terdapat di dalamnya masih bersifat general. Selanjutnya setiap kelas turunan akan mendefinisikan ulang fungsi virtual itu dengan mengisikan perintah-perintah yang sudah spesifik sesuai dengan kebutuhan dari kelas turunan tersebut [36]. Dalam simulator ns-3 fungsi virtual banyak diterapkan. Salah satu contoh dari penggunaan fungsi virtual di dalam simulator ns-3 adalah pada modul QoE Monitor. Coba amati pendefinisian kelas Metric dalam file metric.h, ditunjukkan pada Tabel VI-31.

Tabel VI-31. Definisi kelas Metric.

```
class Metric
  {
  public:
    virtual bool EvaluateQoe
    (std::string originalFilename, std::string receivedFilename) = 0;
    virtual bool PrintResults
    (std::string outputFilename, bool headers) = 0;
  };
```

Pada Tabel VI-31 dapat dilihat penggunaan keyword virtual yang menyatakan bahwa pada kelas metric di definisikan dua buah fungsi virtual yaitu virtual bool EvaluateQoe dan virtual bool PrintResults.

VI.5.2 **Pembaharuan Fungsi (Override)**

Seperti yang telah dikemukakan sebelumnya bahwa fungsi virtual dapat didefinisikan ulang pada kelas turunannya. Hal ini tentu memungkinkan untuk terjadinya proses pembaharuan dari definisi suatu fungsi. Dalam pemrograman berorientasi objek, proses pembaharuan tersebut disebut dengan istilah override. Coba amati kode pada Tabel VI-32.

Tabel VI-32. Override kelas Metric.

```
Class PsnrMetric : public /*ns3::*/Metric
  {
  public:
    PsnrMetric();
    typedef struct MetricRow
    {
      unsigned int m_frameNum;
      double m_psnrY;
      ...
    } MetricRow;
    virtual bool
    EvaluateQoe
    (std::string originalFilename, std::string receivedFilename);
    virtual bool PrintResults
    (std::string outputFilename, bool headers);
     ...
  };
```

Pada Tabel VI-32 terdapat pendeklarasian fungsi virtual pada kelas PsnrMetric yang merupakan kelas turunan dari kelas Metric. Selain kelas PsnrMetric mewarisi sifat-sifat dari kelas Metric terjadi pula proses pembaharuan fungsi (override) di dalamnya. Detil definisi fungsi dapat dilihat pada file psnr-metric.cc.

Tabel VI-33. Implementasi kelas virtual pada SsimMetric.

```
class SsimMetric : public /*ns3::*/Metric
  {
  public:
    SsimMetric();
    typedef struct MetricRow
    {
      unsigned int m_frameNum;
      ...
    } MetricRow;

    virtual bool EvaluateQoe
    (std::string originalFilename, std::string receivedFilename);
    virtual bool PrintResults
    (std::string outputFilename, bool headers);
    ...
  };
```

Pada Tabel VI-33 terdapat pendeklarasian fungsi virtual pada kelas SsimMetric yang merupakan kelas turunan dari kelas Metric. Selain kelas SsimMetric mewarisi sifat-sifat dari kelas Metric terjadi pula proses pembaharuan fungsi (override) di dalamnya. Detil definisi fungsi dapat dilihat pada file ssim-metric.cc.

Penjelasan di atas menunjukkan bahwa pada simulator ns-3 diterapkan juga fungsi virtual yang diperlukan untuk proses poliforfisme.

VI.5.3 **Fungsi Virtual Murni (Pure Virtual Function)**

Kembali pada sub-bab VI.5.1 tentang fungsi virtual, coba amati pendeklarasian dari fungsi virtual EvaluateQoe dan PrintResults pada kelas Metric. Pada proses pendeklarasian fungsi, fungsi virtual diisi dengan nilai 0. Maksud dari itu adalah kita mendeklarasikan fungsi tersebut sebagai fungsi virtual murni (pure virtual function). Dalam pemrograman berorientasi objek fungsi virtual murni adalah fungsi yang murni tidak mempunyai definisi sama sekali [36]. Artinya fungsi tersebut baru didefinisikan pada kelas turunannya. Tabel VI-34 adalah bentuk umum pembuatan sebuah fungsi virtual murni.

Tabel VI-34. Pembuatan fungsi virtual murni.

```
virtual tipe_data nama_fungsi (parameter1, parameter2, ...) = 0;
```

Dalam simulator ns-3, konsep ini telah digunakan pada pendeklarasian fungsi virtual dari kelas Metric. Silahkan amati kode pada Tabel VI-35.

Tabel VI-35. Deklarasi fungsi virtual pada kelas Metric.

```
class Metric
  {
  public:
    virtual bool EvaluateQoe
    (std::string originalFilename, std::string receivedFilename) = 0;
    virtual bool PrintResults
    (std::string outputFilename, bool headers) = 0;
  };
```

Kelas abstrak adalah sebuah kelas yang mempunyai fungsi virtual murni (pure virtual function) di dalamnya. Dalam pemrograman berorientasi objek tidak diijinkan untuk melakukan instansiasi dari kelas abstrak [36]. Coba amati kembali kelas Metric yang terdapat pada file metric.h. Kelas ini merupakan kelas abstrak karena mempunyai fungsi virtual murni.

Tabel VI-36. Deklarasi kelas Metric pada script qoe-monitor.cc.

```
If (enablePsnr)
    {
      /* Computing PSNR */
      std::cout << "PSNR computing...";
      std::cout.flush();
      Metric psnr;
      ...
     }
```

Potongan kode pada Tabel VI-36 terdapat instansiasi dari kelas abstrak, yaitu kelas Metric. Apabila program dijalankan maka akan terdapat error seperti ditunjukan oleh hasil kompilasi pada Tabel VI-37. Hal ini berarti programmer tidak dapat mendeklarasikan objek psnr dengan kelas Metric, karena kelas Metric mempunyai fungsi virtual murni atau kelas abstrak.

Tabel VI-37. Hasil error dari penggunaan kelas abstrak saat kompilasi.

```
../scratch/qoe-monitor-example-1.cc: In function int main(int,
char**):
../scratch/qoe-monitor-example-1.cc:290:14: error:
cannot declare variable psnr to be of abstract type ns3::Metric
./ns3/metric.h:27:9: note:
because the following virtual functions are pure within ns3::Metric:
./ns3/metric.h:31:5: note:
virtual bool ns3::Metric::EvaluateQoe(std::string, std::string)
./ns3/metric.h:34:5: note:
virtual bool ns3::Metric::PrintResults(std::string, bool)
```

VI.5.4 **Polimorfisme (Polymorphism)**

Secara definisi, polimorfisme dapat diartikan sebagai kemampuan mengungkap suatu hal yang berbeda melalui suatu cara yang sama [36]. Perhatikan kembali program di sub bab sebelumnya, di mana pada program tersebut kita tidak dapat melakukan instansiasi pada kelas Metric, sehingga untuk mengakses fungsi EvaluateQoE() dan PrintResults() dilakukan pada kelas-kelas turunannya (PsnrMetric dan SsimMetric).

Pada pemrograman berorientasi objek, terutama C++ didapatkan cara untuk dapat mengakses fungsi-fungsi di dalam kelas turunan hanya dengan mendeklarasikan sebuah pointer yang bertipe kelas induk. Selanjutnya pointer tersebut akan dapat menunjuk ke tipe-tipe kelas turunan yang ada. Untuk mengetahui konsep yang terdapat di dalamnya, perhatikan kode QoE Monitor pada Tabel VI-38.

Tabel VI-38. Konsep polimorfisme.

```
if (enablePsnr)
    {
      /* Computing PSNR */
      std::cout << "PSNR computing...";
      Metric *psnr;
      psnr = new PsnrMetric;
      psnr->EvaluateQoe(rawFilename, receivedRawFilename);
      /* Print the metric output without any header */
      psnr->PrintResults(metricFile.c_str(), false);
      std::cout << " done!\n";
}
```

Apabila program dikompilasi, maka tidak akan terjadi error. Pada potongan kode tersebut (Tabel VI-38), kita mendeklarasikan pointer psnr terhadap kelas Metric (yang merupakan kelas induk), namun kita dapat mengakses fungsi EvaluateQoE() dan PrintResults() yang terdapat di dalam kelas PsnrMetric (yang merupakan kelas turunan).

VI.6 Typecasting dan RTTI

Typecasting adalah merubah suatu variabel atau objek yang memiliki tipe tertentu dengan tipe data lain. Sebagai contoh, dalam program terdapat variabel X yang bertipe int. Namun suatu saat kita dapat merubah variabel ke tipe char. Di sini berarti kita telah melakukan typecasting terhadap variabel X [36].

Dalam bahasa C++ yang digunakan oleh simulator ns-3, masih terdapat 4 macam typecasting yang masing-masing mempunyai fungsi-fungsi spesifik. Adapun keempat macam itu adalah dynamic_cast, const_cast, static_cast, dan reinterpret_cast. Dalam buku ini hanya dibahas penggunaan dynamic_cast karena typecasting tersebut digunakan oleh simulator ns-3.

VI.6.1 Menggunakan dynamic_cast

Typecasting ini dilakukan untuk objek yang bertipe kelas dimana kelas tersebut adalah kelas turunan dari kelas lainnya. Sebagai contoh kita mempunyai kelas B yang merupakan turunan dari kelas A [36]. Dengan menggunakan dynamic_cast, kita dapat menganggap objek dengan tipe A sebagai objek B. Selain itu kita juga dapat menganggap objek B sebagai objek A jika pointer sedang menunjuk objek A. Adapun bentuk umum dari penggunaan dynamic_cast adalah seperti pada Tabel VI-39.

Tabel VI-39. Bentuk umum dynamic_cast.

```
dynamic_cast <T> (ekspresi);
```

T di atas tidak lain merupakan sebuah tipe target yang harus merupakan pointer atau reference. Sedangkan ekspresi di atas merupakan ekspresi yang akan dilakukan cast (dianggap tipe lain). Tabel VI-40 adalah potongan kode dalam script qoe-monitor.cc yang menggunakan dynamic_cast.

Tabel VI-40. Penggunaan dynamic_cast pada script qoe-monitor.cc.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("2Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install(nodes);
PointToPointNetDevice* receiverDevice =
dynamic_cast<PointToPointNetDevice*> (PeekPointer(devices.Get(1)));
```

Coba amati potongan kode pada Tabel VI-40, terjadi proses casting dari objek device yang bertipe kelas NetDeviceContainer terhadap objek receiverDevice yang bertipe kelas PointToPointNetDevice.

VI.6.2 **Run-Time Type Identification (RTTI)**

Bagi yang sebelumnya terbiasa memprogram menggunakan bahasa non-polymorphic, seperti bahasa C, mungkin istilah RTTI merupakan bahasa baru. Namun dalam bahasa polymorphic seperti C++, kita dapat mengenali tipe data dari suatu objek pada saat program dieksekusi (run-time) dengan menggunakan keyword typeid. Adapun bentuk umum dari penggunaan typeid ini adalah seperti pada Tabel VI-41.

Tabel VI-41. Bentuk umum typeid.

```
typeid (nama_objek);
```

Nama_objek yang dimaksud pada Tabel VI-41 adalah semua objek yang terdapat di dalam C++, bisa berupa variabel maupun instance dari suatu kelas.

VI.7 Pemogramman Lanjutan Menggunakan Template

Dalam pemrograman berorientasi objek, terdapat fitur yang disebut dengan template. Dengan menggunakan template, kita dapat membuat fungsi-fungsi atau kelas-kelas generik. Artinya, sekali kita membuat fungsi atau kelas tersebut, maka kita dapat menggunakannya untuk berbagai tipe data tanpa harus mengulang penulisan kode atau kelas tersebut. Dalam C++, template dibuat dengan menggunakan kata kunci **template**. Template dibagi menjadi 2 bagian yaitu template fungsi dan template kelas [36].

VI.7.1 Template Fungsi

Seperti telah disinggung sebelumnya bahwa dengan adanya template, kita dapat membuat fungsi-fungsi generik. Adapun yang dinamakan fungsi generik adalah fungsi yang mempunyai parameter bertipe generik. Artinya, pada saat pemanggilan fungsi tersebut, tipe dari parameter aktual yang dimasukan dapat bersifat dinamis (berubah-ubah). Adapun bentuk umum dari pembuatan fungsi generik (template fungsi) adalah seperti pada Tabel VI-42.

Tabel VI-42. Bentuk umum penggunaan template.

```
Template <class T>
tipe_data nama_fungsi (parameter1, parameter2, ...) {
statment_yang_akan_dilakukan;
...
}
```

Kata kunci template di atas berfungsi untuk menunjukan bahwa fungsi tersebut merupakan fungsi generik. Sedangkan T yang ditulis dalam <class T> disebut dengan tipe generik, yang kemudian akan digunakan dalam pendeklarasian parameter dari fungsi tersebut.

VI.7.2 Template Kelas

Selain menyediakan template untuk fungsi, C++ juga menyediakan template untuk kelas. Sama seperti fungsi generik, kelas generik juga dapat digunakan untuk tipe-tipe data yang berbeda. Pada prinsipnya pendefinisian sebuah template kelas adalah sama dengan pendefinisian kelas biasa, hanya saja variabel yang dideklarasikan di dalamnya bersifat generik. Adapun bentuk umum dari pendefinisian sebuah template kelas (kelas generik) adalah seperti pada Tabel VI-43.

Tabel VI-43. Bentuk umum template kelas.

```
template <class T> class nama_kelas {
akses_specifier:
data_data;
fungsi_fungsi;
...
};
```

Sedangkan proses instansiasi dari template kelas tersebut mempunyai bentuk umum seperti pada Tabel VI-44.

Tabel VI-44. Bentuk umum pembuatan objek dari template.

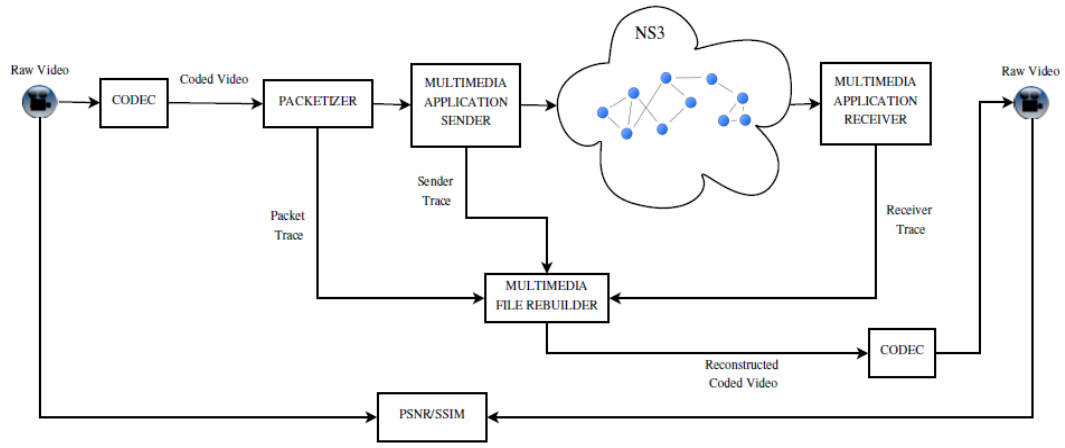
```
nama_kelas <tipe_data> nama_instance;
```

BAB VII

Pengujian Kualitas Layanan Video Menggunakan Tool Simulasi

VII.1 Kerangka Kerja Simulasi untuk Pengukuran Kualitas Video

Penggunaan simulator penting untuk memudahkan kustomisasi parameter-parameter jaringan dan aplikasi untuk mendapatkan target kinerja penjaminan QoS dan QoE, tanpa risiko disrupsi apabila dilakukan pada jaringan riil operasional. Salah satu contoh modul simulator yang mengintegrasikan simulasi jaringan dan pengukuran kualitas video adalah modul open-source QoE-Monitor yang dikembangkan oleh University of Modena di Italia [37] [38]. Pengembangan modul tersebut termotivasi oleh pesatnya penggunaan simulator discrete-event ns-3 [29] dan tool pengujian kualitas EvalVid [39] [40]. Gambar VII-1 menunjukkan kerangka kerja evaluasi dengan QoE-Monitor [37], di mana cloud jaringannya adalah jaringan yang disimulasikan dengan ns-3, dan fungsi-fungsi dari modul QoE-Monitor berperan untuk pemrosesan packet data dan konten video, di pengirim dan penerima, dan kemudian melakukan analisa kualitas.



Gambar VII-1. Kerangka Kerja Simulasi [37].

Filosofi pengujian kualitas video pada Gambar VII-1 adalah menggunakan full-reference metrics yang didasarkan atas perbedaan atau variasi antara original (reference) video dan video diterima (dalam hal ini turut menyertakan faktor codec). Untuk itu, pada kerangka kerja perlu diperhatikan bagian terkait pemrosesan codec video (transmit dan receive), bagian jaringan pada ns3, dan pada bagian interface antar bagian-bagian tersebut. QoE-monitor menyediakan modul PSNR dan SSIM untuk

pengukuran kualitas video, dan utilisasi ffmpeg [41] untuk pemrosesan codec video, seperti kompresi, decompresi, pembuatan container, dan sebagainya.

Dari Gambar VII-1, terlihat bahwa dimungkinkan untuk mengembangkan QoE-Monitor agar menyertakan fitur-fitur yang lebih lengkap untuk optimalisasi kinerja jaringan yang menjamin QoS/QoE layanan multimedia. Salah satu model korelasi QoS/QoE lainnya yang telah diadopsi oleh industri adalah standard ITU-T G.1070 yang tergolong no-reference metrics. Standard ini tergolong model parametric planning yang ideal digunakan untuk perencanaan sistem dan jaringan multimedia, sebagaimana dijelaskan di BAB III. Untuk implementasi model ini pada ns-3, diperlukan suatu probe untuk mengukur packet loss untuk frame-frame video yang dilewatkan pada ns-3, dan informasi video bit rate dan frame rate di sini penerima. Ekstensi ns-3 dan QoE-Monitor untuk menyertakan model pengujian kualitas video berbasiskan standard ITU-T G.1070 merupakan kontribusi baru, dan dipublikasikan oleh penulis di [42]. Pada BAB ini dipaparkan kustomisasi ns-3 dan QoE-Monitor untuk adopsi standard ITU-T G.1070, uji coba penggunaannya untuk dibandingkan dengan PSNR dan SSIM.

VII.2 Tool QoE Monitor

QoE Monitor adalah modul pada simulator ns-3 yang digunakan untuk mengevaluasi kualitas video yang diterima oleh pengguna layanan, dengan mengadopsi formula dari metrik obyektif berdasarkan parameter-parameter yang ada di dalam jaringan. Evaluasi kualitas video dengan full-reference metric dapat dilakukan dengan melihat perbedaan-perbedaan (atau variasi) dari dua buah file video yang dijalankan, yaitu membandingkan antara video original (reference) dan video yang diterima. QoE-Monitor dirancang dan diimplementasikan dalam Sistem Operasi Linux, yang terbentuk menjadi modul di dalam ns-3, yang ditulis menggunakan bahasa pemogramman C++, dan mereproduksi komponen-komponen utama dari tool EvalVid.

Mengacu ke Gambar VII-1, prinsip kerja QoE-Monitor dapat dijelaskan sebagai berikut. Di sisi pengirim, sumber video referensi dalam format raw (uncompressed) di-coded ke format compressed digital menggunakan tool ffmpeg, dan kemudian dipaketisasi menggunakan Packetizer. Tahapan ini juga membuat packet trace berisikan packet ID, ukuran packet, dan timestamp. Packet yang dihasilkan kemudian dikirim oleh Multimedia Application Sender ke jaringan yang disimulasikan oleh ns-3. Packet yang dikirim ke jaringan telah disusun dalam format yang menyertakan RTP (Real-Time Protocol), dan tersusun dalam aliran UDP datagram. Packets yang dikirim ke jaringan dicatat dalam sender trace, bersama dengan packet ID dan timestamp.

Di sisi penerima, packet diterima oleh Multimedia Application Receiver yang mengekstrak informasi dari header di tiap-tiap paket yang diterima dan membuat file receiver trace berisikan paket ID dan timestamps. File packet trace, sender trace, dan receiver trace digunakan oleh Multimedia File Rebuilder untuk merekontruksi ulang video, dan melaporkan kemungkinan terjadinya error pada file video. Kualitas dari file video yang direkontruksi ulang tergantung pada berapa banyak packet yang berhasil diterima, dan ini dipengaruhi oleh parameter QoS jaringan seperti packet delay, jitter, dan error rate.

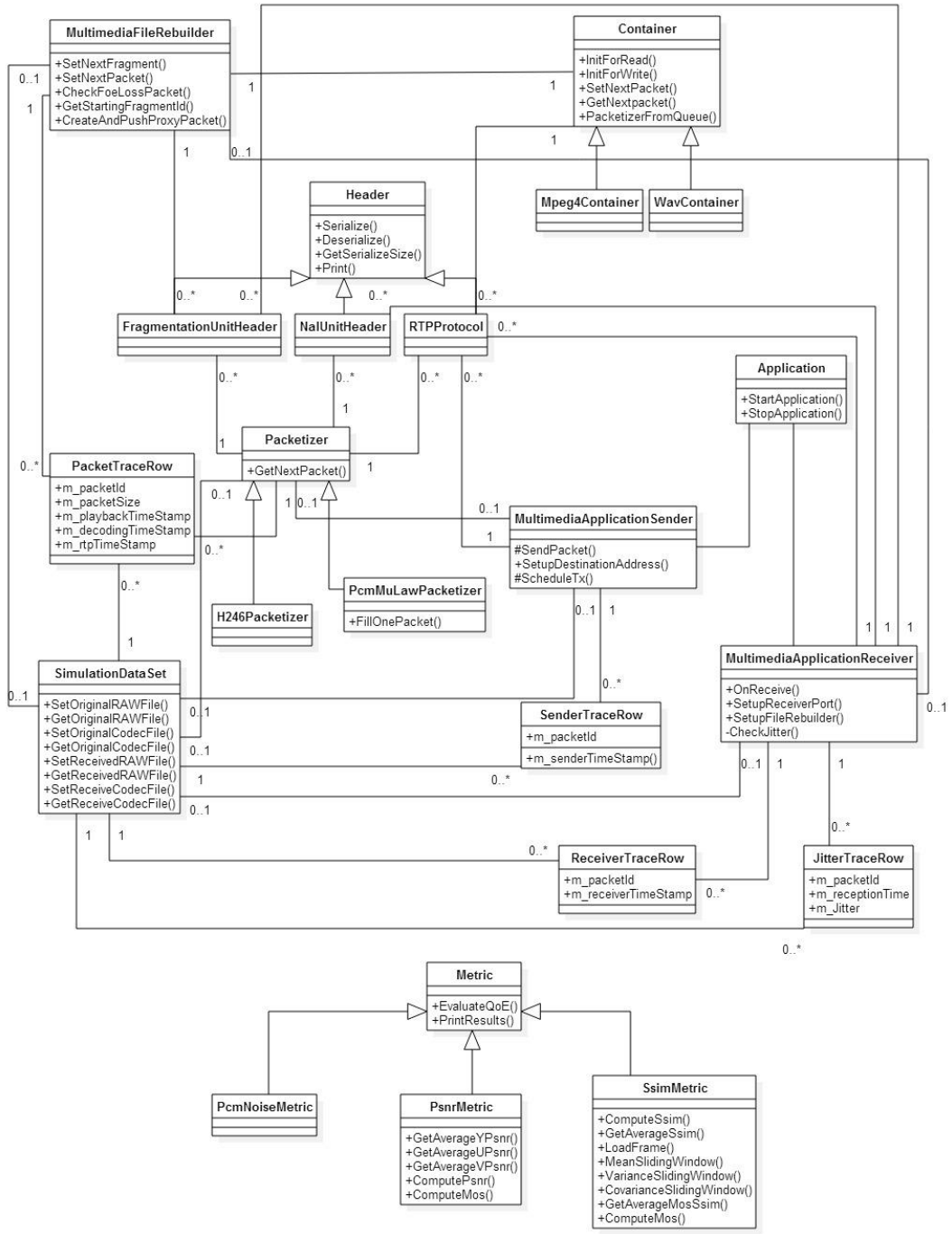
Selanjutnya, video hasil rekontruksi di-decode untuk membentuk satu file raw video di sisi penerima. Setelah itu, file raw video di sisi penerima dibandingkan dengan file video referensi dengan menggunakan salah satu metrik perhitungan kualitas video, contohnya menggunakan PSNR atau SSIM. Dengan tool simulasi ini pengguna dapat melihat efek dari video yang ditransmisikan, codec yang digunakan, parameter QoS jaringan atas kualitas video yang diterima.

VII.2.1 **Desain Class QoE-Monitor di NS-3**

Pada bagian ini akan dijelaskan kelas utama yang membentuk QoE Monitor pada NS-3. Dengan menggunakan bantuan UML (Unified Modelling Language) class diagram dapat dibuat suatu hirarki class dan hubungan antar class satu dan lainnya. Hubungan antar class di dalam modul QoE Monitor dapat dilihat pada Gambar VII-2.

Salah satu class penting dalam modul QoE-Monitor adalah class SimulationDataset, yang menyimpan semua variable yang dibutuhkan saat menjalankan simulasi, karena menyimpan semua nama file multimedia yang dibutuhkan untuk evaluasi, dan semua file trace yang dibentuk oleh kejadian-kejadian yang berkaitan dengan proses transmisi. Saat modul QoE-Monitor bekerja dengan konten multimedia, terdapat beberapa class yang dirancang untuk memanipulasi file audio dan video. Virtual class bernama Container dan semua class yang diturunkan darinya (contoh: Mpeg4Container, WavContainer) adalah class yang memproses file multimedia (contoh: Mpeg4, Wave). Konten multimedia yang ditransmisikan melalui jaringan yang disimulasikan diproses oleh class MultimediaApplicationSender, yang diturunkan dari class Application yang disediakan oleh kerangka kerja ns-3. Proses pengiriman dibuat standar untuk menyembunyikan hal-hal detil, dan dibentuk kelas virtual Packetizer, yang berfungsi untuk menyembunyikan proses paketisasi di aplikasi. Modul QoE-Monitor menggunakan transmisi berbasis RTP, dan di framework ns-3 telah diimplementasikan fitur dasar dari protokol RTP yang dirancang secara sederhana pada class RTPProtokol. Pada class ini dapat dibuat paket

header dengan informasi berupa paket ID dan timetamps ke sisi penerima seperti yang telah dijelaskan sebelumnya.



Gambar VII-2. Class Diagram QoE-Monitor.

Modul QoE-Monitor baru mendukung codec H.264 [22] [43] yang juga terintegrasi dengan dengan ffmpeg [41], dan telah mengimplementasikan Network Abstraction Layer (NAL) packet header. Class NALUnitHeader adalah class yang memproses unit header

dari codec. Karena NAL bisa lebih besar dari ukuran MTU di jaringan, maka dibuat penambahan class yang akan melaporkan ke penerima bahwa packet yang diterima sesungguhnya merupakan satu fragment dari packet panjang, yang dalam hal ini diproses oleh class FragmentationUnitHeader.

Informasi yang berguna terhadap masing-masing packet yang di-generate oleh paketizer tersimpan di file trace paket yang didefinisikan oleh kelas SimulationDataset. File trace packet berisikan informasi paket ID, ukuran paket, timestamp palyback, timestamp decoding, dan timestamp RTP. Setelah terisi kemudian akan di-push ke dalam struktur paket trace. Setelah membentuk packet RTP dari packetizer, lalu packet akan dikirim melalui jaringan. Informasi yang berhubungan dengan masing-masing packet yang dikirim adalah paket ID, dan waktu pengiriman yang dibangun oleh struktur SenderTraceRow, yang dimasukan ke struktur trace sender berlokasi di class SimulationDataset.

Penerima diimplementasikan oleh class MultimediaApplicationReceiver. Setelah packet diterima, class MultimediaApplicationReceiver akan melakukan perhitungan estimation jitter dan memeriksa packet mengkuti ketentuan RFC 3550, untuk menguji apakah packet data sampai pada waktu yang tepat atau tidak. Untuk packet yang berhasil sampai ke penerima, informasi jitter akan disimpan didalam struktur JitterTraceRow yang berisi paket ID, waktu penerimaan dan nilai estimated jitter dari paket. Kemudian struktur JitterTraceRow akan di-push ke file trace jitter yang ada di SimulationDataset. Terakhir, packet-paket yang berhasil diterima akan diproses oleh class MultimediaFileRebuilder, yang membentuk file hasil transmisi. Proses rebuild mungkin saja terjadi korupsi jika dibandingkan dengan file yang dikirim, hal ini dapat terjadi karena adanya packet loss saat transmisi dan proses coding.

Berikut ini ringkasan dari class pada QoE-Monitor, diturunkan dari ns-3 sehingga untuk pendefinisiannya dibuat pada **namespace ns3**:

- **SimulationDataSet**: Berisikan variabel-variabel yang menyimpan state dari simulasi yang sedang berjalan, seperti menyimpan semua file-file multimedia yang digunakan untuk evaluasi, semua traces oleh event-event yang terkait dengan transmisi data, dsb.
- **Container**: Ini adalah virtual class, yang kemudian dapat diturunkan menjadi class container yang lebih spesifik seperti Mpeg4Container dan WavContainer untuk penggunaan pada file-file dengan format Mpeg4 dan Wav.
- **MultimediaApplicationSender**: Class ini diturunkan dari class Application yang disediakan oleh ns3, dan bertugas sebagai aplikasi multimedia pengirim paket pada kerangka kerja ns3.

- **Packetizer**: Pure virtual class Packetizer yang menyembunyikan detil proses packetization ke application, dan detil packetization dilakukan oleh class-class yang lain atau turunannya.
- **RtpProtocol**: Class ini mengimplementasikan fitur-fitur RTP (Real-Time Protocol) pada packet-packet yang dilewatkan pada jaringan ns3.
- **NalUnitHeader**: Class untuk implementasi NAL (Network Abstraction Layer) untuk mendukung codec H.264. Class yang terkait adalah **FragmentationUnitHeader** yang digunakan untuk melaporkan ke receiver kalau packet yang diterima berasal dari satu fragment yang lebih panjang dari network MTU.
- **MultimediaApplicationReceiver**: Class ini diturunkan dari class Application yang disediakan oleh ns3, dan bertugas sebagai aplikasi multimedia penerima paket pada kerangka kerja ns3.
- **MultimediaFileRebuilder**: Bertugas untuk menyusun kembali (rebuild) file yang ditransmisikan. Perlu diingat bahwa packet yang dikirim bisa hilang karena proses transmisi dan codec, sehingga untuk evaluasi perbandingan antara file yang diterima dengan yang dikirim, perlu dilakukan penyesuaian (align) atas paket-paket yang diterima.
- **PsnrMetric**: Bertugas untuk menghitung nilai PSNR antara video yang diterima dan referensinya.
- **SsimMetric**: Bertugas untuk menghitung nilai index SSIM, yang menyertakan faktor korelasi spatial sehingga lebih akurat daripada PSNR.

VII.3 Pengukuran Kualitas Video Menggunakan PSNR dan SSIM

QoE-Monitor menyediakan metrics PSNR [15] dan SSIM [16] untuk pengukuran kualitas video, karena populer digunakan pada komunitas ilmiah skala laboratorium, di mana file-file video di sisi pengirim dan penerima tersedia untuk dibandingkan. Pada QoE-Monitor, PSNR diimplementasikan pada class PsnrMetric yang menghitung nilai PSNR antara reference video dengan video yang diterima dan direkontruksi kembali. Nilai PSNR dihitung dengan membandingkan signal-to-noise ratio tiap frame dari kedua file video (reference dan diterima) Nilai dari metric PSNR berkisar dari 0 sampai infinite (dalam modul QoE-Monitor diasumsikan nilai tertinggi adalah 99). Semakin tinggi nilai PSNR, diasumsikan kualitas yang lebih bagus, walau sebagaimana dijelaskan di Bab III bahwa PSNR hanya membandingkan error antar dua gambar tanpa memperhitungkan fitur persepsi manusia (*human perception*).

Metric yang menyertakan karakteristik persepsi manusia adalah SSIM, yang berusaha untuk ekstrak struktur informasi dari gambar atas dasar pixel-pixel yang berdekatan secara spatial berkorelasi tinggi dan saling berketergantungan. Komputasi Metric ini diimplementasikan pada class SsimMetric dari QoE-Monitor. Nilai SSIM terletak diantara 0 dan 1, dimana semakin tinggi mengindikasikan kualitas yang lebih baik. Konsekuensi untuk menyertakan karakteristik persepsi manusia ini menyebabkan semakin beratnya komputasi dari metric SSIM.

QoE-Monitor menyediakan opsi untuk enable atau disable PSNR dan SSIM sebagai mana ditunjukkan pada Tabel VII-1 dan Tabel VII-2.

Tabel VII-1. Enable atau disable PSNR.

```
      if (enablePsnr)
      {
        /* Computing PSNR */
        std::cout << "PSNR computing...";
        std::cout.flush();
        PsnrMetric psnr;
        psnr.EvaluateQoe(rawFilename, receivedRawFilename);
        /* Print the metric output without any header */
        psnr.PrintResults(metricFile.c_str(), false);
        std::cout << " done!\n";
      }
```

Tabel VII-2. Enable atau disable SSIM.

```
  if (enableSsim)
      {
        /* Computing SSIM */
        std::cout << "SSIM computing...";
        std::cout.flush();
        SsimMetric ssim;
        ssim.EvaluateQoe(rawFilename, receivedRawFilename);
        /* Print the metric output without any header */
        ssim.PrintResults(metricFile.c_str(), false);
        std::cout << " done!\n";
      }
```

Sebagaimana dijelaskan di atas, nilai metric PSNR dan SSIM masih dalam format mereka masing-masing. Untuk dapat melakukan komparasi antar metric, termasuk dengan hasil dari standard ITU-T G.1070, diperlukan pemetaan ke format MOS. Pemetaan ini diperoleh dari hasil pengujian subjective terkini, yang telah dipaparkan di BAB II (lihat Tabel II-2). Implementasi pemetaan pada modul simulator dilakukan dengan cara mendefinisikan member variable MOS pada class PsnrMetric dan SsimMetric, dan

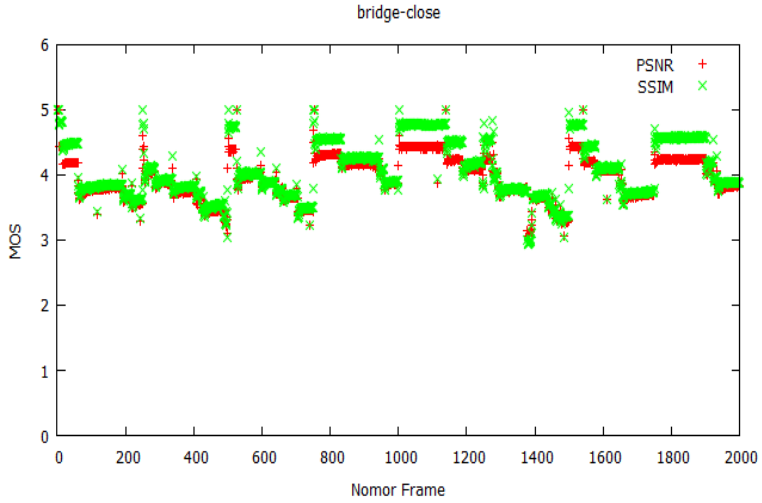
pendefinisian private function untuk menghitung nilai MOS. Tabel VII-3 menggambarkan deklarasi class PsnrMetric yang menyertakan pemetaan ke MOS.

Uji coba pemetaan dilakukan dengan menggunakan file video bridge-close_cif (jumlah frame = 2000), topologi point-to-point dengan packet loss = 0.005. Detil skenario simulasi akan dijelaskan pada sub-bab berikutnya. Perbandingan nilai PSNR dan SSIM dalam format MOS ditunjukkan pada Gambar VII-3, dan terlihat adanya konsistensi fluktuasi nilai MOS antara PSNR dan SSIM, dan terlihat hasil PSNR lebih pesimistik dibandingkan SSIM. Juga pola memburuknya kualitas video terlihat seragam, di mana hal ini menunjukkan dampak random packet loss dengan distribusi uniform pada rentang frame 1-2000.

Tabel VII-3. Class PsnrMetric yang menyertakan pemetaan ke MOS.

```
      class PsnrMetric : public /*ns3::*/Metric
      {
            public:
            PsnrMetric();
            typedef struct MetricRow
            {
                        unsigned int m_frameNum;
                        double m_psnrY;
                        double m_psnrU;
                        double m_psnrV;
                        double m_mosY;
                        double m_mosU;
                        double m_mosV;
            } MetricRow;
            double GetAverageMosY();
            double GetAverageMosU();
            double GetAverageMosV();

            private:
            double m_avgMosY;
            double m_avgMosU;
            double m_avgMosV;
            double ComputeMos(const double);
            };
      }
      //Compute MOS for PSNR based on mapping in Zinner's paper
      double PsnrMetric::ComputeMos(const double psnr)
      {
            double mos;
            mos = psnr >= 45 ? 5 : psnr >= 33 ? 4 : psnr >= 27.4 ? 3 :
psnr >= 18.7 ? 2 : 1;
            return(mos);
      }
```



Gambar VII-3. MOS vs nomor frame menggunakan PSNR dan SSIM.

VII.4 Penggunaan Standard ITU-T G.1070 pada QoE-Monitor

Tahapan implementasi standard ITU-T G.1070 pada QoE-Monitor mencakup revisi source code eksisting qoe-monitor yang menyertakan class baru untuk ITU-T G.1070 metric, dan uji coba simulasi pada topologi sederhana menggunakan parameter jaringan dan sumber video yang mendekati kondisi riil. QoE-Monitor eksisting memiliki beberapa kekurangan, di antaranya class PsnrMetric dan SsimMetric belum menyediakan pemetaan nilai PSNR dan SSIM ke format MOS (Mean Opinion Score). Revisi pemetaan ini telah dijelaskan di sub-bab sebelumnya. Dalam hal resolusi, PsnrMetric dan SsimMetric eksisting hanya dapat digunakan untuk video dengan resolusi cif, sehingga source code perlu direvisi untuk probing variable width dan height dari AVContext dari library libavcodec pada ffmpeg.

VII.4.1 Desain Class dan Revisi Modul

Class Metric pada QoE-Monitor dirancang untuk full-reference metric yang menggunakan informasi file-file source dan destination video. QoE-Monitor belum menyediakan class tersendiri untuk melakukan coding dan decoding, dan tahapan ini dilakukan oleh ffmpeg yang dijalankan via shell. Untuk desain class, dirancang class NRMetric yang memerlukan file video yang diterima, dan network QoS sebagai input parameter untuk evaluasi QoE. Class ini bisa digunakan untuk menurunkan class untuk

metode lainnya yang tergolong no-reference metric, seperti standard ITU-T G.1070. Snapshop code untuk class NR_Metric ditunjukkan pada Tabel VII-4

Tabel VII-4. Snapshot untuk class NR_Metric.

```
namespace ns3
{
  class NR_Metric
  {
  public:
    virtual bool
    EvaluateQoe(std::string receivedFilename,SimulationDataset*
dataset) = 0;
    virtual bool PrintResults(std::string outputFilename, bool
headers) = 0;
  };
}
```

Implementasi dari ITU-T G.1070 disertakan pada class ITUG1070Metric yang diturunkan dari class NR_Metric. Fungsi untuk menghitung MOS diimplementasikan sebagai member function dari class ITUG1070, sedangkan fungsi untuk penyimpanan data mengacu ke fungsi yang serupa pada class PsnrMetric dan SsimMetric. Sebagaimana dipaparkan pada BAB III, perhitungan MOS menurut standard ITU-T G.1070 memerlukan tiga parameter masukan, yaitu video bit rate, video frame rate, dan packet loss. Masukan untuk video rate, dan video frame rate bergantung pada tool compression yang digunakan, dan dienkapsulasi oleh class Container beserta turunannya. Pada riset yang dilakukan penulis digunakan video dalam format mp4, dan informasi video bit rate dan frame rate disertakan pada turunan dari class ns3::Container. Snapshop untuk class ITUG1070Metric ditunjukkan pada Tabel VII-5.

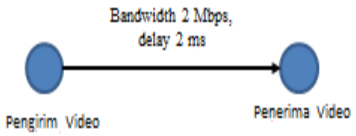
Beberapa class lainnya yang direvisi dalah class SimulationDataset untuk menyertakan struktur frame-trace, class packetizer, dan mpeg4-packetizer untuk menyertakan informasi frame_id dan packet_id, yang bersama-sama dengan packet_trace dan receive_trace digunakan untuk menghitung packet loss per frame video.

Tabel VII-5. Snapshot class ITUG1070Metric.

```
namespace ns3
{
  class ITUG1070Metric : public NR_Metric
  {
  public:
    ITUG1070Metric();
    typedef struct MetricRow
    {
      unsigned int m_frameId;
      double       m_bitRate;  // Brv
      double       m_frameRate; // Frv
      double       m_packetLost; //Pplv
      double       m_mos;
    } MetricRow;
    Bool EvaluateQoe(std::string
receivedCodedFilename,SimulationDataset *dataset);
    Bool PrintResults(std::string originalCodedFilename,bool headers);
    Int GetAverageBitRate();
    Int GetAverageFrameRate();
    Int GetAveragePacketLoss();
    double GetAverageMos();
    double ComputeMOS(double, double, double);
  private:
    unsigned int m_frameNumTot;
    double m_avgBitRate;
    double m_avgFrameRate;
    double m_avgPacketLost;
    double m_avgMos;
    std::vector<MetricRow> m_metric;
    // the following are parameters required for coefficient #5 for
ITU-T G.1070 Codec Information
    static const double v1=5.517, v2=1.29E-2, v3=3.459, v4=178.53,
v5=1.02, v6=1.15, v7=3.55E-4, v8=0.114, v9=513.77, v10=0.736, v11=-
6.451, v12=13.684;
       double Brv, Frv,Pplv;
    double Dpplv,DFrv,Iofr,Ofr,Icoding,Vq;
  };
}
ITUG1070Metric::ComputeMOS(double Brv, double Frv, double Pplv)
  {
    Dpplv = v10 + v11*exp(-(Frv/v8)) + v12*exp(-(Brv/v9));
    DFrv = v6 + v7*Brv;
    Iofr = v3 - (v3/(1+pow(Brv/v4,v5)));
    if ((Iofr < 0)|| (Iofr > 4))
    {
        std::cout << "Error:   Iofr di luar rentang nilai" <<
std::endl;
          exit(2);
    }
    Ofr = v1 + v2*Brv;
    if (Ofr > 30.0) Ofr = 30.0;
    else if (Ofr < 1.0) Ofr = 1.0;
    Icoding = Iofr * exp(-((log(Frv)-log(Ofr))*(log(Frv)-
log(Ofr)))/(2.0*DFrv*DFrv));
    Vq = 1 + Icoding*exp(-(Pplv/Dpplv));
    return Vq;
  }
```

VII.4.2 **Skenario Simulasi dan Uji Coba**

Skenario simulasi adalah pada jaringan point-to-point seperti pada Gambar VII-4, di mana tiap node menggunakan IP protocol (topologi serupa digunakan di [37]). Bandwidth dari point-to-point link adalah 2 Mbps dan delay 2 ms. MTU dari packet yang dikirim adalah 1400, dan packet error rate menggunakan distribusi uniform dengan nilai rata-rata yang divariasikan, yaitu 0.001, 0.005, 0.01, 0.05. Untuk sumber video, dipilih beberapa reference video [44, 45], dengan jumlah frame lebih besar dari 1500. Tabel VII-6 menunjukkan reference video yang digunakan. Pada tahapan simulasi, dilakukan perbandingan atas penggunaan standard ITU-T G.1070 dengan metric PSNR dan SSIM, menggunakan format yang sama, yaitu MOS (Mean Opinion Score).

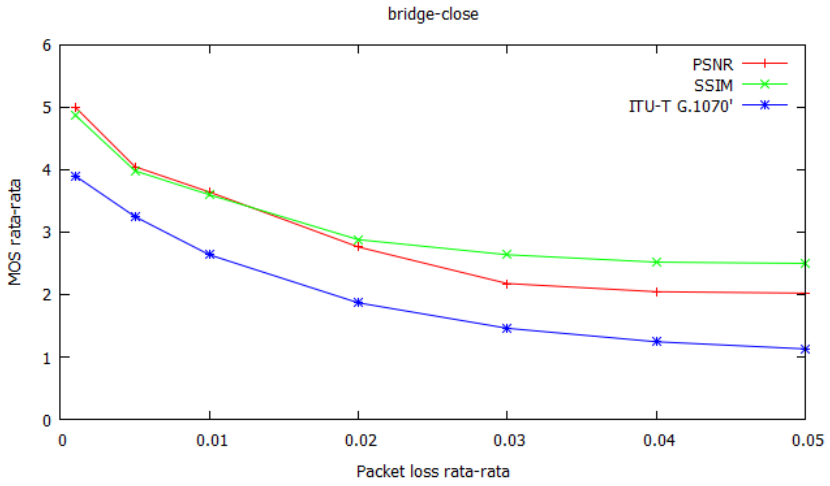


Gambar VII-4. Topologi Jaringan Point-to-Point.

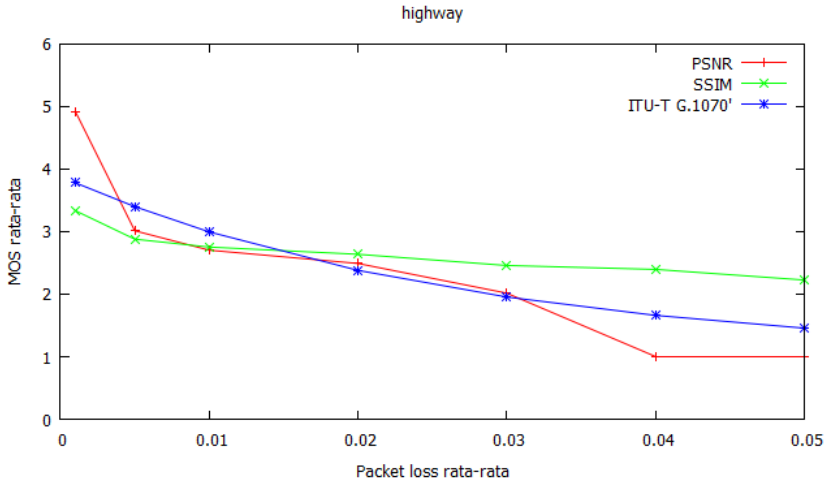
Tabel VII-6. Reference Video.

	Jumlah Frame	Bit Rate Rata-Rata (kbps)	Frame Rate
Bridge_close	**2000**	**1435**	**25**
Highway	**1832**	**2006**	**25**

Gambar VII-5 dan Gambar VII-6 menunjukkan nilai rata-rata MOS yang diperoleh pada packet loss rata-rata dan video rate menggunakan sumber video dengan bit rate yang berbeda. Terlihat bahwa secara rata-rata PSNR lebih pesimistik dari SSIM, dan semakin memburuk dengan meningkatnya packet loss. Dampak packet loss pada PSNR makin terlihat untuk penggunaan video bit rate yang lebih besar, dan dalam hal ini bit rate Highway yang mendekati kapasitas link.



Gambar VII-5. MOS vs Packet Loss (bridge-close).



Gambar VII-6. MOS vs Packet Loss (highway).

Perhitungan MOS dengan ITU-T G.1070 menggunakan nilai rata-rata dari video bit rate, video frame rate, dan percentage dari packet loss, sebagai cara yang umum digunakan untuk perencanaan jaringan. Terlihat bahwa perubahan nilai MOS dari ITU-T berkorelasi cukup baik dengan hasil dari PSNR dan SSIM, dan estimasi oleh ITU-T G.1070 lebih baik untuk pada Highway (bit rate lebih tinggi).

Yang menjadi catatan di sini untuk penggunaan ITU-T G.1070, estimasi MOS tidak mempertimbangkan karakteristik konten dari kualitas video. Hal ini terlihat dari deviasi nilai MOS dari ITU-T G.1070 dengan nilai MOS dari SSIM pada Gambar VII-5 dan Gambar VII-6. Namun dengan waktu komputasi yang jauh lebih cepat, standard ITU-T G.1070 ideal digunakan untuk simulasi perencanaan dan rekonfigurasi jaringan

multimedia. Untuk penyempurnaan lebih lanjut, perlu dikaji penggunaan versi penyempurnaan (*enhanced*) dari standard ITU-T G.1070 [25] [20] [21].

Contoh penggunaan standard ITU-T G.1070 pada ns-3 dan QoE-Monitor menunjukkan fleksibilitas dari tool open-source untuk desain dan optimalisasi jaringan multimedia yang menyertakan model korelasi QoS/QoE. Beberapa topik riset lanjutan yang pembaca dapat eksplorasi, untuk penyempurnaan hasil yang dilaporkan di [42], adalah sebagai berikut:

- Penggunaan standard ITU-T G.1070 untuk estimasi MOS dari traffic trace (pcap) pada simulasi jaringan. Untuk monitoring MOS pada traffic trace jaringan riil, standard ITU-T G.1070 telah digunakan pada tool proprietary RadVision [46].
- Investigasi dampak variasi parameter MTU, bandwidth, dan buffer size pada packet error rate, dan konsekuensinya pada estimasi MOS.
- Investigasi dampak mixed traffic dan variasi distribusi random error pada jalur komunikasi terhadap estimasi MOS.
- Uji coba penggunaan standard ITU-T G.1070 untuk simulasi pada jaringan dengan topologi yang lebih realistik, menyertakan aggregasi traffic dan potensi congestion pada node.
- Mengembangkan skenario simulasi yang lebih realistik di sisi aplikasi yang digunakan oleh pengguna, contoh penggunaan BoxingExperince [47] yang mensimulasikan streaming beberapa clients pada sebuah komputer yang menggunakan teknologi virtualisasi.

DAFTAR PUSTAKA

[1] A. S. Mohamed, Automatic Evaluation of Real-Time Multimedia Quality : A Neural Network Approach, 2003.

[2] A. Takahashi, D. Hands and V. Barriac, "Standardization Activities in The ITU for a QoE Assessment of IPTV," *IEEE Communications Magazine,* pp. 78-84, February, 2008.

[3] M. Fiedler, T. Hossfeld and P. Tran-Gia, "A Generic Quantitative Relationship Between Quality of Experience and Quality of Service," *IEEE Network,* vol. 24, no. 2, pp. 36-41, March, 2010.

[4] M. Fiedler and T. Hossfeld, "Quality of Experience-Related Differential Equation and Provisioning-Delivery Hysteresis," in *Proc. 21st ITC Specialist Seminar on Multimedia*, 2010.

[5] M. Alreshoodi and J. Woods, "Survey on QoE\QoS Correlation Models for Multimedia Services," *International Journal of Distributed and Parallel Systems (IJDPS),* vol. 4, no. 3, pp. 53-72, May, 2013.

[6] T. Zinner, T. Hossfeld, T. N. Minash and M. Fiedler, "Controlled vs Uncontrolled Degradation of QoE: The Provisioning-Delivery Hysteresis in Case of Video," in *Proc. New Dimesiona in the Assessment and Support of Quality of Experience (QoE) for Multimedia Applications*, June, 2010.

[7] T. Zinner, T. Hossfeld, T. N. Minhas and M. Fiedler, "Controlled vs Uncontrolled Degradations of QoE: The Provisioning-Delivery Hysteresis in Case of Video," in *Proc. 3rd Workshop on Quality of Experience for Multimedia Content Sharing (QoEMCS)*, Tampere, 2010.

[8] L. Skorin-Kapov and M. Varela, "A Multi-Dimensional View of QoE: The ARCU Model," in *Proc.IEEE MIPRO*, 2012.

[9] Y.-C. Chang, C.-J. Chang, K.-T. Chen and C.-L. Lei, "Radar Chart: Scanning for High QoE in QoS Dimensions," in *Proc. IEEE Communications Quality and Reliability (CQR)*, 2010.

[10] G. O'Driscoll, Next Generation IPTV Services and Technologies, Wiley, 2007.

[11] S. Winkler, Digital Video Quality: Vision Models and Metrics, Wiley, 2005.

[12] ITU-T P.910: Subjective Video Quality Assessment Methods for Multimedia Applications, 2008.

[13] ITU-T J.144: Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

[14] "Qpsnr A Quick PSNR/SSIM Analyzer for Linux," [Online]. Available: http://qpsnr.youlink.org/.

[15] S. Winkler and P. Mohandas, "The Evolution of Video Quality Measurement: from PSNR to Hybrid Metrics," *IEEE Transaction on Broadcasting,* vol. 54, no. 3, pp. 1-9, 2008.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing,* vol. 13, no. 4, pp. 600-612, 2004.

[17] T. Zinner, O. Abboud, O. Hohlfeld and T. Hossfeld, "Towards QoE Management for Scalable Video Streaming," in *Proc. 21th ITC Specialist Seminar on Multimedia Applications – Traffic, Performance and QoE*, 2010.

[18] ITU-T G.1070: Opinion Model for Video-Telephony Applications, 2012.

[19] K. Yamagishi, T. Tominaga, T. Hayashi and A. Takahashi, "Objective Quality Evaluation Model for Videophone Services," *NTT Technical Review,* vol. 5, no. 6, pp. 1-5, 2007.

[20] J. Joskowicz and J. Carlos Lopez Ardao, "Enhancements to the Opinion Model for Video-Telephony Applications," in *Proc. 5th International Latin American Networking Conference*, 2009.

[21] J. Joskowicz, J. Carlos Lopez Ardao and R. Sotelo, "Quantitative Modeling of the Impact of Video Content in the ITU-T G.1070 Video Quality Estimation Function," *Informatica Educacao: teoria & practica,* vol. 14, no. 2, pp. 191-203, 2011.

[22] ITU-T H.264: Advanced video coding for generic audiovisual services, 2007.

[23] N. D. Narvekar, T. Liu, D. Zou and J. A. Bloom, "Extending G.1070 for Video Quality Monitoring," in *Proc. IEEE Multimedia and Expo*, 2011.

[24] B. Wang, D. Zou, R. Ding, S. Bhagavathy, N. Narverkar and J. Bloom, "Efficient Frame Complexity Estimation and Application to G.1070 Video Quality Monitoring," in *Proc. Workshop on Quality of Multimedia Experience (QoMEX)*, 2011.

[25] T. Liu, N. Narvekar, B. Wang, R. Ding, D. Zou, G. Cash, S. Bhagavathy and J. Bloom, "Real-Time Video Quality Monitoring," *EURASIP Journal on Advances in Signal Processing,* 2011.

[26] J. Gross and M. Gunes, "Introduction," in *Modeling and Tools for Network Simulation*, Springer, 2010, pp. 1-11.

[27] J. Banks, J. S. Carson II, B. L. Nelson and D. M. Nicol, Discrete-Event System Simulation, Prentice-Hall, 2009.

[28] G. F. Riley and T. R. Henderson, "The ns–3 Network Simulator," in *Modeling and Tools for Network Simulation*, Springer, 2010, pp. 15-34.

[29] "NS-3 Network Simulator," [Online]. Available: http://www.nsnam.org.

[30] "ns-3 Tutorial Release ns-3.15," [Online]. Available: http://www.nsnam.org/documentation/older/.

[31] GNU, "GNU coding standards, online-manual," [Online]. Available: http://www.gnu.org/prep/standards.

[32] G. F. Riley, "The Georgia Tech Network Simulator," in *Proc. of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 2003.

[33] M. Lacage and T. R. Henderson, "Yet Another Network Simulator," in *Proc. Workshop on ns-2: the IP Network Simulator (WNS2)*, 2006.

[34] "Wireshark," [Online]. Available: http://www.wireshark.org.

[35] K. C. Chen, "Comparison of object-oriented and procedure-based computer language: case study of c++ programming," *Journal of Computer Information Systems,* vol. 5, no. 1, pp. 70-76, 2004.

[36] B. Raharjo, Pemogramman C++ revisi kedua, Bandung: Informatka, 2011.

[37] D. Saladino, A. Paganelli and C. Maurizio, "A Tool for Multimedia Quality Assessment in NS3: QoE Monitor," *Simulation Modelling Practice and Theory,* vol. 32, pp. 30-41, March, 2013.

[38] "QoE-Monitor," [Online]. Available: http://www.sourceforge.net/projects/ns3qoemonitor/.

[39] "EvalVid – A Video Quality Evaluation Tool-Set," [Online]. Available: http://www.tkn.tu-berlin.de/research/evalvid/.

[40] J. Klaue, B. Rathke and A. Woliz, "Evalvid - A Framework for Video Transmission and Quality Evaluation," in *Proc. 13th International conference on Modelling Techniques and Tools for Computer Performance Evaluation* , 2003.

[41] "Ffmpeg," [Online]. Available: http://www.ffmpeg.org.

[42] A. A. N. A. Kusuma and D. Irawan, "Penggunaan Standard ITU-T G.1070 untuk Estimasi Quality of Experience Layanan Video pada Simulasi Jaringan," in *Prosiding Seminar Insentif Riset SINas (INSINAS 2014)*, Bandung, 2014.

[43] "x264 H.264/AVC encoder," [Online]. Available: http://www. videolan.org/developers/x264.html.

[44] "H264 YUV CIF EvalVid Reference Videos," [Online]. Available: www2.tkn.tu-berlin.de/research/evalvid/cif.html.

[45] "Video Quality Expert Group," [Online]. Available: www.vqeg.org.

[46] Radvision, "VQ Monitor User Guide Version 2.7," 2012.

[47] J. Bustos-Jimenez, R. Alonso, C. Faundez and H. Meric, "BoxingExperience: Measuring QoS and QoE of Multimedia Streaming Using NS3, LXC, and VLC," in *Proc. IEEE Workshop on Network Measurements (WNM)*, Edmonton, 2014.

BIOGRAFI PENULIS

Anak Agung Ngurah Ananda Kusuma, lahir di Denpasar 19 Juli 1969, bekerja sebagai Perekayasa Madya (*Specialist Engineer*) di Pusat Teknologi Informasi dan Komunikasi (PTIK), Badan Pengkajian dan Penerapan Teknologi (BPPT). Penulis menyelesaikan pendidikan S1 dengan predikat *First Class Honours* dari University of Tasmania, S2 dari Royal Melbourne Institute of Technology, dan S3 dari University of Melbourne. Selama menempuh studi, penulis memperoleh beasiswa Science and Technology Manpower Development Program (STMDP) dari pemerintah Indonesia dan dari Australian Agency for Industrial Development (AusAID). Penulis memiliki minat penelitian dan kerekayasaan yang terkait peningkatan kinerja QoS dan QoE dengan pemanfaatan sumber daya yang efisien pada jaringan nirkabel, jaringan berbasis IP, dan sistem terdistribusi. Saat ini penulis menjabat sebagai Group Leader WBS 2 Pemanfaatan Protocol *Advanced* untuk Jaringan Multimedia pada Kegiatan Multimedia Digital Networks (MDN) di PTIK-BPPT, dosen paruh waktu, dan menjadi anggota PII dan Communication Society dari IEEE.

Dedy Irawan, lahir di Jakarta 25 Oktober 1982, bekerja sebagai Perekayasa Pertama (*Engineer*) di Pusat Teknologi Informasi dan Komunikasi (PTIK), Badan Pengkajian dan Penerapan Teknologi (BPPT). Penulis menyelesaikan pendidikan S1 dari Universitas Budi Luhur. Selama menempuh studi, penulis aktif dalam kegiatan laboratorium Jaringan Komputer. Penulis memiliki minat penelitian dan kerekayasaan terkait dengan jaringan komputer, pemrograman lanjutan untuk *hacking* kernel, pemodelan dan pemrograman simulator jaringan, dan pemrograman protokol jaringan. Saat ini penulis menjabat sebagai Leader WP 2.2 Pengkajian Metode Estimasi QoE (Korelasi QoE/QoS) yang merupakan sub kegiatan dari WBS 2 Pemanfaatan Protocol *Advanced* untuk Jaringan Multimedia pada Kegiatan Multimedia Digital Networks (MDN) di PTIK-BPPT.

PUSAT TEKNOLOGI INFORMASI DAN KOMUNIKASI

[PTIK]

BADAN PENGKAJIAN DAN PENERAPAN TEKNOLOGI

[BPPT]

ISBN 978-979-26-3020-6