OPEN ACCESS
Indo-JC
Indonesian Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

IND. JOURNAL ON COMPUTING
VOL. 1, ISSUE. 1, MARCH 2016. PP. 25-36
DOI:10.21108/INDOJC.2016.1.1.34

Pengembangan dan Analisa Enkoder dan Dekoder Protocol Data Unit (PDU) pada Wireless interoperable Microwave Access (WIMAX)

Aji Gautama Putrada [#1]

 # Program Studi S1 Informatika, Telkom University

Jl. Telekomunikasi No. 1, Bandung, Indonesia

[1] ajigps@telkomuniversity.ac.id

Abstract

PDU Encoder and Decoder is a part of the MAC layer of WiMAX that has the task of forming the data to be sent and that has been received. PDU Encoder and Decoder is an integral part of WiMAX and WiMAX can not run without it. If WiMAX is implemented with SoC, the MAC can be made as a software that runs on top of the SoC device. The purpose of this research is to develop and analyze The PDU Encoder and Decoder software. To create such software, Software Engineering is used, and the design process is waterfall. The software is written in C programming language and is tested with Unit Tests and Functional Tests. Unit Test and Functional Test is conducted to ascertain whether this software can be installed on such WiMAX device. Results from this research is the implementation of software Encoder and Decoder PDU successfully went through Unit Tests and Functional Tests. Implying that this software can be implemented and tested over WiMAX communication devices.

Keywords: WiMAX, PDU, Encoder, Decoder, MAC, Packing, Fragmentation, CRC, Software Engineering, Unit Test

Abstraksi

PDU Enkoder dan Dekoder merupakan bagian dari MAC Layer pada WiMAX yang bertugas membentuk data yang akan dikirim dan telah diterima. PDU Enkoder dan Dekoder merupakan bagian integral dari WiMAX dan tanpanya WiMAX tidak dapat berjalan. Jika WiMAX dibuat dengan SoC, maka MAC dapat dibuat sebagai software yang berjalan di atas perangkat SoC tersebut. Tujuan dari research ini adalah mengembangkan dan menganalisa software Enkoder dan Dekoder PDU. Rekayasa Perangkat Lunak ini menggunakan proses desain waterfall. Softwarenya ditulis dengan bahasa C dan diuji dengan Unit Test dan Fungsional Test. Unit Test dan Fungsional Test dilakukan untuk memastikan apakah software ini dapat dipasang pada Perangkat WiMAX. Hasil dari penelitian ini adalah implementasi perangkat lunak Enkoder dan Dekoder PDU berhasil melalui Unit Test dan Fungsional Test. Sehingga perangkat lunak ini dapat diimplementasikan dan diuji di atas perangkat komunikasi WiMAX.

Keywords: WiMAX, PDU, Enkoder, Dekoder, MAC, Packing, Fragmentation, CRC, Rekayasa Perangkat Lunak, Unit Test

RECEIVED ON FEBRUARY 2016. ACCEPTED ON MARCH 2016

I. PENDAHULUAN

PDU Enkoder dan Dekoder merupakan bagian dari MAC Layer pada WiMAX yang bertugas membentuk data yang akan dikirim dan telah diterima. PDU Enkoder dan Dekoder merupakan bagian integral dari WiMAX dan tanpanya WiMAX tidak dapat berjalan. Jika WiMAX dibuat dengan SoC [1], maka MAC dapat dibuat sebagai software yang berjalan di atas perangkat SoC tersebut. Tujuan dari research ini adalah mengembangkan dan menganalisa software Enkoder dan Dekoder PDU. Rekayasa Perangkat Lunak ini menggunakan proses desain waterfall. Softwarenya ditulis dengan bahasa C dan diuji dengan Unit Test dan Fungsional Test. Unit Test dan Fungsional Test dilakukan untuk memastikan apakah software ini dapat dipasang pada Perangkat WiMAX dan apakah data yang dihasilkan sesuai dengan Standar WiMAX (IEEE 802.16-2012) [2].

Bentuk dari MAC PDU tergantung dari layanan yang diterima data pada MAC Layer. Tidak semua layanan akan dibahas pada penelitian ini. Pada penelitian ini layanan yang akan dibahas adalah:

1) Packing
2) Fragmentation
3) Cyclic Redundancy Check (CRC)

II. MAC PDU

Setiap Layer Stack pada OSI Layer mempunyai data menuju ke Layer bawahnya saat sedang *transmit* dan data menuju Layer atasnya saat sedang *receive*. Setiap data yang menuju Layer bawahnya, atau keluar dari layer tersebut untuk di-*transmit,* dinamakan Protocol Data Unit (PDU). Sebaliknya, setiap data yang menuju Layer atasnya, atau keluar dari layer tersebut pada saat *receive*, dinamakan Service Data Unit (SDU). Pada *transmitter* WiMAX data diterima dari layer network [2]. Data dari layer network dinamakan Packet, Setelah Packet melalui *Convergence Sublayer* [3 – sitasi perlu dicari] Packet dinamakan Service Data Unit (SDU). SDU ini melalui berbagai layanan yang disediakan oleh MAC. Pengolahan data yang dialami SDU akan membuat SDU ini berubah menjadi PDU. Segala perubahan yang dialami oleh PDU disimpan di dalam PDU Header. Sehingga ketika PDU dikirim dan diterima di receiver WiMAX penerima. PDU tersebut dapat diubah kembali menjadi SDU yang identik dengan SDU yang dikirim.

A. Layanan pada MAC

Gambar 1 di bawah menggambarkan format MAC PDU Header [2]. Dari gambar tersebut dapat dilihat layanan-layanan pada MAC yang mempengaruhi suatu data yang melewati MAC Layer WiMAX.

		Bit							
		0	1	2	3	4	5	6	7
Byte	0	HT	EC	Type					
	1	ESF	CI	EKS		Rsv	LEN (MSB)		
	2	LEN (LSB)							
	3	CID (MSB)							
	4	CID (LSB)							
	5	HCS							

Gambar 1. Format MAC PDU Header

Gambar di atas menjelaskan bahwa MAC Header terdiri dari 6 Byte. Dalam 6 Byte tersebut ada 8 *field* (tidak termasuk Rsv yang artinya Reserved atau tidak digunakan). Berikut merupakan penjelasan masing-masing *field*:

1) *HT:* Header Type. Panjang 1 bit, artinya ada dua jenis Header: General MAC Header (bernilai 0) dan Bandwidth Request Header (bernilai 1). Penelitian ini akan membahas General MAC Header saja.

2) *EC:* Encryption Control. Panjang 1 bit, jika bernilai 0, maka PDU tidak melalui enkiripsi. Sebaliknya jika bernilai 1, maka data melalui enkripsi. Enkripsi tidak dibahas pada penelitian ini.

3) *Type:* Panjang 6 bit. Satu bit merepresentasikan kehadiran satu layanan pada PDU. Berarti total ada 6 layanan yang dijelaskan pada *field* Type ini: Rsv, ARQ Feedback Payload Present, (Fragmentation dan Packing Subheader) Extended Type, Fragmentation Subheader Present, Packing Subheader Present, FAST-FEEDBACK Allocation Subheader Present (Uplink)/Grant Management Subheader (Downlink). Pada penelitian ini yang dibahas hanya field Fragmentation Subheader Present dan Packing Subheader Present saja.

4) *ESF:* Extended Subheader Field. Mempunyai Panjang 1 bit. Jika bernilai 1, ada Extended Subheader setelah Header. ESF tidak dibahas di penelitian ini.

4) *CI:* CRC Indicator. Panjang 1 bit. Jika nilainya 0 berarti tidak ada CRC Checksum pada PDU. Jika bernilai 1 berarti ada CRC Checksum pada PDU.

5) *EKS:* Encryption Key Sequence. EKS ini hanya akan berisi nilai yang bermanfaat jika Enkripsi digunakan atau EC = 1. EKS tidak dibahas pada penelitian ini.

6) *LEN:* Length. Total panjang LEN adalah 11 bit. Mengindikasikan panjang dari PDU (Sudah termasuk MAC Header dan CRC) dalam . Jika LEN mengindikasikan panjang maksimal PDU berarti PDU paling panjang adalah $2^{11} – 1$ Byte = 2043 Byte

7) *CID:* Connection Identifier. WiMAX bersifat connection oriented. Artinya WiMAX memberi ID perproses jika ada suatu proses ingin melakukan pengiriman dalam jaringan WiMAX. ID tersebut dinamakan CID. Panjangnya adalah 16 bit. Artinya dalam suatu jaringan WiMAX maksimum ada 2^{16} koneksi = 65.536 koneksi.

8) *HCS:* Header Check Sequence. MAC Header melalui CRC dengan format CRC-8 [4]. Checksum dari CRC ini disimpan dalam HCS.

Nilai HT, Type, CI, LEN, CID, dan HCS akan sangat penting pada penelitian ini karena terlibat dalam proses Packing, Fragmentation dan CRC.

B. Packing dan Fragmentation

Packing secara harfiah berarti mengumpulkan, dan Fragmentation berarti memotong. Packing dan Fragmentation pada WiMAX digunakan untuk menyesuaikan ukuran SD) dengan ukuran yang dialokasikan oleh Scheduler. Scheduler memberi alokasi ukuran sesuai dengan kondisi koneksi yang ada.

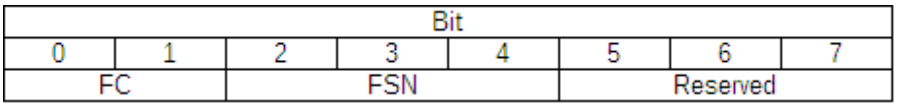
Jika data tersebut lebih kecil dari ukuran yang dialokasikan, data tersebut dapat digabungkan dengan data-data lain yang memiliki CID yang sama, proses ini dinamakan Packing. Jika Packing terjadi pada suatu PDU, maka *field* Type pada MAC Header akan bernilai 0x02.

Di Receiver, program PDU Dekoder akan memeriksa MAC Header sehingga mengetahui adanya *pack*, dengan informasi dari PSH, masing-masing *pack* dapat dipisahkan dari MAC PDU. Proses ini diberi nama Unpacking.

Jika data tersebut lebih besar dari ukuran yang dialokasikan, data akan dipotong. Proses ini dinamakan Fragmentation. Jika Fragmentation terjadi pada suatu PDU, maka *field* Type pada MAC Header akan bernilai 0x04. Karena proses Fragmentation ini diindikasikan dalam FSH maka PDU Dekoder dapat mendeteksi bahwa isi dari MAC Header adalah *fragment*. Sehingga *fragment-fragment* ini akan dikumpulkan di PDU

Dekoder. Setelah semua *fragment* terkumpul, akan dilakukan proses Reassembly untuk menyatukan mereka kembali.

Setiap Packing dan Fragmentation akan disertai dengan Packing Subheader (PSH) dan Fragmentation Subheader (FSH). Gambar 2 dan 3 berturut-turut menjelaskan format dari FSH dan PSH.

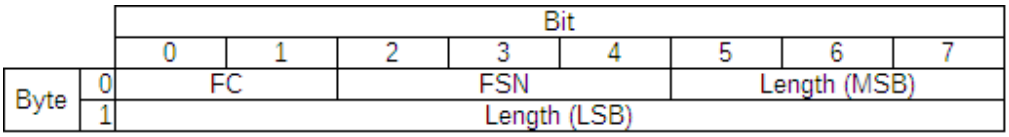


Gambar 2. Format Fragmentation Subheader

FSH mempunyai panjang 1 Byte dan terdiri dari 2 *field*. Penjelasan *field* pada Gambar 2:

1) *FC:* Mengindikasikan status dari *fragment* tersebut. 00 berarti bukan *fragment*, 01 *fragment* terakhir, 10 berarti akhir *fragment* pertama, 11 berarti *fragment* tengah-tengah.

2) *FSN:* Fragment Serial Number. Memberikan nomor urut kepada potongan tersebut.



Gambar 3. Format Packing Subheader

PSH mempunyai panjang 2 Byte dan terdiri dari 3 *field*. Selain Length, field lain mempunyai nama, kegunaan, ukuran dan posisi yang sama dengan FSH. Penjelasan *field* pada Gambar 3:

1) *FC:* Mengindikasikan status dari *fragment* tersebut. 00 berarti bukan *fragment*, 01 *fragment* terakhir, 10 berarti akhir *fragment* pertama, 11 berarti *fragment* tengah-tengah.

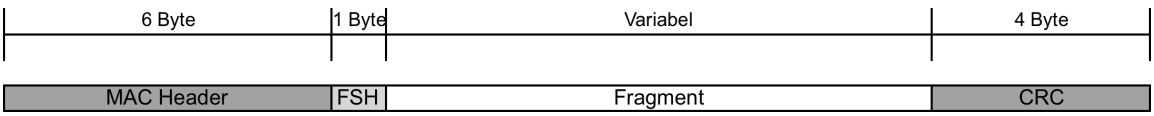
2) *FSN:* Fragment Serial Number. Memberikan nomor urut kepada potongan tersebut.

3) *Length:* Panjang dari *pack* dalam Byte. Panjang total nya adalah 11 bit. Ini berarti panjang maksimum dari suatu *pack* adalah 2^{11} – 1 Byte = 2047 Byte.

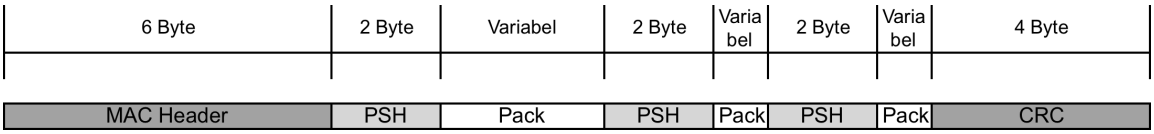
C. CRC

Jika CI pada MAC PDU Header bernilai 1 (lihat bagian B), data akan mengalami CRC. CRC adalah mekanisme *error checking*. Data yang melalui mekanisme CRC akan diberi CRC Checksum di belakang data tersebut. CRC Checksum ini digunakan pada *receiver* untuk melihat apakah data tersebut *error* atau tidak. Jika *error* Checksum yang melalui proses kalkulasi CRC akan bernilai ≠ 0. Kemudian data yang *error* akan dibuang oleh Receiver (Rx). Jika tidak *error*, Checksum akan bernilai = 0. MAC WiMAX menggunakan CRC-32. Artinya Checksum mempunyai panjang 32 bit (4 Byte).

Gambar 4 dan Gambar 5 berturut-turut memberi gambaran mengenai format PDU yang mempunyai CRC dan berisi *fragment* dan PDU yang mempunyai CRC dan berisi *pack*.



Gambar 4. MAC PDU berisi 1 *fragment* dan CRC



Gambar 5. MAC PDU berisi 3 *pack* dan CRC

III. PERANCANGAN PDU ENKODER DAN DEKODER

Penelitian ini bertujuan mengembangkan dan menganalisis PDU Enkoder dan Dekoder. Rekayasa perangkat lunak PDU Enkoder dan PDU Dekoder menggunakan model proses *waterfall* [5]. Di mana pengembangan terdiri dari proses Analisis Kebutuhan → Desain → Implementasi → Pengujian.

Analisis Kebutuhan dari perangkat lunak PDU Enkoder adalah:

1) PDU Enkoder dapat melakukan proses Packing
2) PDU Enkoder dapat melakukan proses Fragmentation
3) PDU Enkoder dapat melakukan proses Enkoding MAC PDU Header
4) PDU Enkoder dapat melakukan proses Kalkulasi CRC

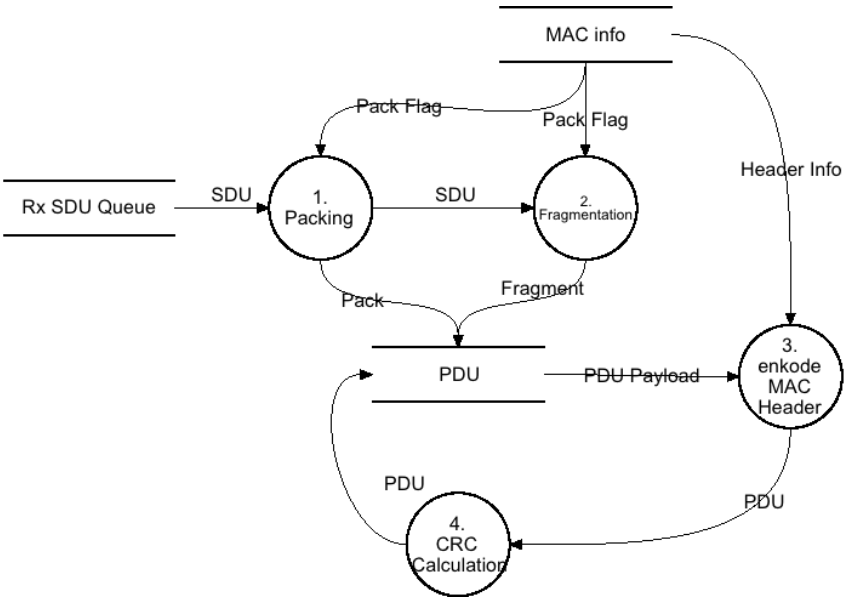
Analisis Kebutuhan dari perangkat lunak PDU Dekoder adalah:

1) PDU Dekoder dapat melakukan proses Error Checking
2) PDU Dekoder dapat melakukan proses Reassembly
3) PDU Dekoder dapat melakukan proses Unpacking

Perancangan perangkat lunak PDU Enkoder dan PDU Dekoder dibuat untuk memenuhi Analisis Kebutuhan di atas. Juga perlu diingatkan bahwa semua proses dan Enkoding harus sesuai dengan Standar WiMAX (IEEE.802.16-2012) [2].

A. Perancangan Encoder

Gambar 6. merupakan DFD dari proses keseluruhan PDU Enkoder. Proses tersebut sudah mencakup semua proses berdasarkan Analisis Kebutuhan PDU Enkoder, yaitu proses Packing, proses Fragmentation, proses Enkoding MAC PDU Header, dan proses Kalkulasi CRC. Bagian berikutnya akan membahas Flowchart dari masing-masing proses.



Gambar 6. DFD Proses PDU Enkoder

1) Proses Packing

Pertama-tama proses Packing akan mendeteksi apakah Packing berlaku untuk CID dari SDU tersebut. Jika Tidak berlaku, proses ini akan berakhir. Jika berlaku, proses Packing akan memasukkan SDU ke dalam MAC PDU Payload. Jika MAC SDU lebih besar dari alokasi panjang MAC PDU Payload, maka SDU akan melalui proses Fragmentation. Jika tidak SDU akan dimasukkan langsung ke dalam MAC PDU Payload, disertai dengan sebuah PSH. Jika masih ada ruang dan masih ada SDU, proses Packing akan diulang kembali.

2) Proses Fragmentation

Pertama-tama proses Fragmentation akan mendeteksi apakah Packing berlaku untuk CID dari SDU tersebut. Jika berlaku, proses ini akan berakhir. Jika tidak, proses Fragmentation akan memasukkan SDU ke dalam MAC PDU Payload. Jika MAC SDU lebih besar dari alokasi panjang MAC PDU Payload, maka SDU akan melalui proses Fragmentation. Jika tidak SDU akan dimasukkan langsung ke dalam MAC PDU Payload, disertai dengan sebuah FSH.

3) Proses Enkoding MAC PDU Header

Proses Enkoding akan memasukkan nilai HT, Type, CI, LEN, dan CID ke dalam MAC PDU Header sesuai dengan format MAC PDU Header yang berlaku. Terakhir akan ada proses menghitung HCS atau Header Check Sequence untuk dimasukkan ke dalam 1 Byte terakhir dari Header.

Di Receiver MAC WiMAX Penerima, sebelum MAC Header dibaca, proses Header Errror Checking akan memeriksa HCS untuk mengetahui apakah Header error atau berisi informasi yang bermanfaat.

4) Proses Kalkulasi CRC

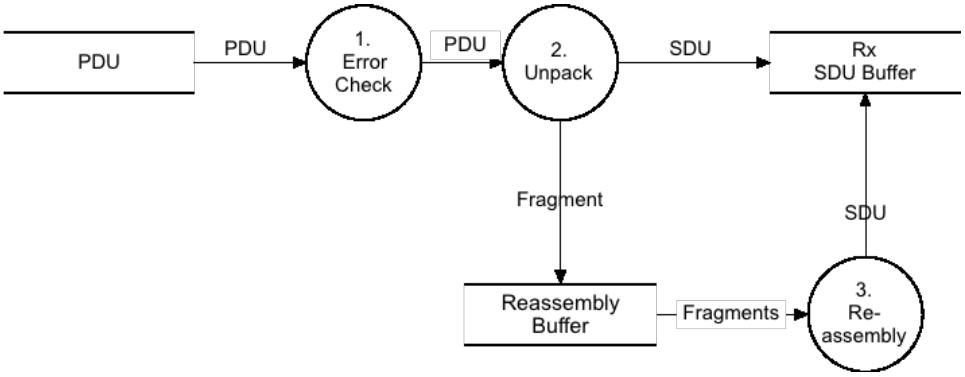
Pertama proses ini akan memeriksa Field CI dalam MAC PDU Header. Jika bernilai 0, proses ini akan berakhir. Jika bernilai 1, proses akan menambah LEN (Length) pada MAC PDU Header dengan 4.

Kalkulasi CRC pada MAC WiMAX menggunakan format CRC-32 [6]. Caranya adalah seluruh PDU (Header dan Payload) akan di-XOR dengan Divisor standard yaitu 0x04C11DB7, dimulai dari nilai 1 pertama

dari PDU. Remainder dari XOR itu akan di-XOR lagi dengan Divisor dimulai dari nilai 1 pertama dari Remainder. Dan selanjutnya untuk Remainder-Remainder berikutnya. Hasil dari perhitungan ini adalah 4 Byte yang diseut sebagai CRC Checksum. 4 Byte ini akan disimpan di 4 Byte terakhir dari PDU.

B. Perancangan PDU Decoder

Gambar 7. merupakan Flowchart dari proses keseluruhan PDU Dekoder. Proses tersebut sudah mencakup semua proses berdasarkan Analisis Kebutuhan PDU Dekoder, yaitu proses Error Checking, proses Unpacking, dan proses Reassembly. Bagian berikutnya akan membahas Flowchart dari masing-masing proses.



Gambar 7. DFD Proses PDU Dekoder

1) Proses Error Checking

Pertama-tama proses ini akan memeriksa HCS dari MAC PDU Header dari PDU ini. Jika ada Error, MAC PDU ini akan dibuang. Jika tidak ada Error, proses ini akan membaca CI (CRC Indicator) pada MAC PDU Header. Jika bernilai 0, proses ini akan berakhir. Jika bernilai 1, proses akan melakukan kalkulasi CRC (identik dengan CRC Calculation pada PDU Enkoder). Jika hasil dari kalkulasi ini adalah 0, berarti tidak ada Error dan proses akan berakhir. Jika hasil dari kalkulasi ini bukan 1, berarti ada Error dan PDU akan dibuang, lalu proses akan berakhir.

2) Proses Unpacking

Pertama-tama proses ini akan memeriksa apakah PDU bernilai Null atau tidak. Jika Null, Proses ini akan berakhir. Jika tidak, proses akan membaca apakah ada FSH Present atau ada PSH Present atau tidak ada keduanya dari MAC PDU Header.

Jika MAC PDU tidak ada FSH Present maupun PSH Present, maka proses Unpacking akan mengambil SDU dalam PDU Payload kemudian memasukkannya ke dalam Rx SDU Queue. Selanjutnya proses akan berakhir.

Jika MAC PDU ada FSH Present, maka proses Unpacking akan mengambil Fragment, Fragment State (awal SDU, akhir SDU, atau tengah-tengah SDU), dan FSN (Fragment Sequence Number) dari Fragment tersebut. Informasi Fragment State, dan FSN dapat diambil dari FSH. Semua informasi ini akan dimasukkan ke dalam Reassembly Buffer. Selanjutnya proses akan berakhir.

Jika MAC PDU ada PSH Present, maka proses Unpacking akan membaca PSH Pertama dan mengambil Pack Pertama. Jika Pack berisi SDU maka SDU akan dimasukkan ke dalam Rx SDU Queue. Jika Pack berisi Fragment, proses akan memasukkan Fragment, Fragment State, dan FSN ke dalam Reassembly Buffer. Proses

akan membaca PSH berikutnya menggunakan Length dari Pack sebelumnya. Proses akan diulang seperti mengambil Pack Pertama. Jika Pack sudah habis, proses akan berakhir.

3) Proses Reassembly

Proses akan membaca Fragment State dari Fragment terakhir pada Reassembly Buffer. Jika State menyatakan bahwa Fragment terakhir bukan Akhir SDU, proses akan berakhir. Jika State menyatakan bahwa Fragment terakhir adalah Akhir SDU, proses akan mengeluarkan semua Fragment dari Reassembly Buffer kemudian melakukan Reassembly (menggabungkan Fragment-Fragment menjadi SDU). SDU ini akan dimasukkan ke dalam Rx SDU Queue.

Pada bagian selanjutnya, pembahasan mengenai implementasi akan memperlihatkan bahwa masing-masing proses yang dijelaskan sebelumnya akan diejawantahkan dalam bentuk fungsi program.

TABEL I
LIST FUNGSI PDU ENKODER DAN PDU DEKODER

No.	Nama Fungsi	Argumen	Tipe Data	Return Value	Tipe Data
PDU Enkoder					
1	Packing	MAC_info	struct	void	void
		Tx_SDU_queue	unsigned char string array		
		PDU	unsigned char string		
		Allocation_length	integer		
2	Fragmentation	MAC_info	struct		
		Tx_SDU_queue	unsigned char string array	void	void
		PDU	unsigned char string		
		Allocation_length	integer		
3	encodeMACPDUHeader	MAC_info	struct	void	void
		PDU	unsigned char string		
4	calculateCRC	PDU	unsigned char string	void	void
PDU Dekoder					
1	errorChecking	PDU	unsigned char string	result	integer
2	Unpacking	PDU	unsigned char string	void	void
		Rx_SDU_queue	unsigned char string array		
		Reassembly_buffer	unsigned char string array		
3	Reassembly	Reassembly_buffer	unsigned char string array	void	void
		Rx_SDU_queue	unsigned char string array		

IV. IMPLEMENTASI DAN PENGUJIAN

TABEL I berisi list Fungsi hasil Implementasi perangkat lunak PDU Enkoder dan PDU Dekoder dibuat dengan bahasa pemrograman C. Implementasi dibuat berdasarkan perancangan pada Bagian III.

TABEL II
LIST ASSERT/TEST VECTOR PDU ENKODER

Nama Fungsi/ Nomor Assert	**Masukan**				**Keluaran yang diharapkan**
Packing	**Packing Allowed**	**Tx_SDU_queue**		**Allocation Length**	**PDU**
		SDU	**Fragment no.**		
Assert 1	Yes	0xAAAAAA	0	23 Byte	0xXXXXXXXXXXXX0005A AAAAA0005BBBBBB8003 CCXXXXXXXX
		0xBBBBBB	0		
		0xCCCCCC	0		
Assert 2	Yes	0xCCCC	1	22 Byte	0xXXXXXXXXXXXX4804C CCC0083DDXXXXXXXX
		0xDDDDDD	0		
		0xEEEE	0		
		0xFF	0		
Assert 3	No	0x1111	0	21 Byte	0xXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXXXX
		0x222222	0		
Fragmentation	**Packing Allowed**	**Tx_SDU_Queue**		**Allocation Length**	**PDU**
		SDU	**Fragment no.**		
Assert 1	No	0xFFFFFFFFFF FFFFFFFFF	2	13 Byte	0xXXXXXXXXXXXXD8FFF FXXXXXXXX
Assert 2	No	0xEEEEEE	8	14 Byte	0xXXXXXXXXXXXX78EEE EEEXXXXXXXXX
Assert 3	Yes	0x33333333	1	20 Byte	0xXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXX
		0x44444444	2		
encodeMACPDUHeader	**MAC_info**				**PDU**
Assert 1	Packing Present = 1 Fragmentation Present = 0 CRC Indicator = 1 Length = 13 CID = 40				0x02400D0028B5XXXXX XXXXXXXXX
Assert 2	Packing Present = 0 Fragmentation Present = 1 CRC Indicator = 1 Length = 500 CID = 600				0x0441F4025860XXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
Assert 3	Packing Present = 0 Fragmentation Present = 0 CRC Indicator = 0 Length = 2047 CID = 65535				0x0007FFFFFF6DXXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
calculateCRC	**PDU**				**PDU**
Assert 1	0x00400D0045ABCDEF0123456789				0x0040110045ABCDEF0 1234567897DCAA585
Assert 2	0x0040160045AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA				0x00401A0045AAAAAAA AAAAAAAAAAAAAAAAAAA

Nama Fungsi/ Nomor Assert	Masukan	Keluaran yang diharapkan
		AAAAAAAA8CFE9238
Assert 3	0x000008AAF712	0x000008AAF712

Unit Test dilakukan untuk menguji kesesuaian Input dan Output masing-masing Fungsi[7]. Yang dimaksud dengan kesesuaian Input dan Output adalah Output yang diberikan oleh suatu Input sesuai dengan Requirement Fungsi tersebut. Jika Unit Test sudah dilewati tanpa kegagalan (Failed) maka Fungsi-Fungsi tersebut dapat digabungkan untuk melakukan pengujian berikutnya yaitu Fungsional Test.

Satu fungsi akan melewati satu Unit Test di mana masing-masing Unit Test akan melakukan 3 kali Assert [8] masing-masing Assert sudah diberikan Test Vector. Test Vector tersebut sudah sesuai dengan standar WiMAX 802.16-2012. Tabel II dan Tabel III berturut-turut memberikan informasi mengenai nilai-nilai Assert masing-masing Unit Test pada PDU Enkoder dan PDU Dekoder.

TABEL III
LIST ASSERT/TEST VECTOR PDU DEKODER

<table>
<tr><th>Nama Fungsi/ Nomor Assert</th><th colspan="3">Masukan</th><th colspan="2">Keluaran yang diharapkan</th></tr>
<tr><td>errorCheck</td><td colspan="3">PDU</td><td colspan="2">PDU</td></tr>
<tr><td>Assert 1</td><td colspan="3">0xABCDE20AAABBBCCC8015299B</td><td colspan="2">0xABCDE20AAABBBCCC8015299B</td></tr>
<tr><td>Assert 2</td><td colspan="3">0xABCDE20AAABB00CC8015299B</td><td colspan="2">NULL</td></tr>
<tr><td>Assert 3</td><td colspan="3">0xAB5DE20AAABBBCCC8015299B</td><td colspan="2">NULL</td></tr>
<tr><td>Unpack</td><td colspan="3">PDU</td><td>Rx_SDU_queue</td><td>Reassembly_buffer</td></tr>
<tr><td>Assert 1</td><td colspan="3">0x80400ABBBBCCABCDEABCDE</td><td>0xABCDEABCDE</td><td>NULL</td></tr>
<tr><td>Assert 2</td><td colspan="3">0x844012CCCC33F81234567ABCD</td><td>NULL</td><td>Fragment: 0x1234567
Fragment State: Tengah SDU
FSN: 7</td></tr>
<tr><td rowspan="2">Assert 3</td><td colspan="3" rowspan="2">0x824016AAAA557003AA0004BBBB8005 CCCCCC</td><td rowspan="2">0xBBBB</td><td>Fragment: 0xAA
Fragment State: Akhir SDU
FSN: 6</td></tr>
<tr><td>Fragment: 0xCC
Fragment State: Awal SDU
FSN: 0</td></tr>
<tr><td rowspan="2">Reassembly</td><td colspan="3">Reassembly_buffer</td><td colspan="2" rowspan="2">Rx_SDU_queue</td></tr>
<tr><td>Fragment</td><td>FSN</td><td>Status</td></tr>
<tr><td>Assert 1</td><td>0xA
0xB
0xC</td><td>0
1
2</td><td>10
11
01</td><td colspan="2">0xABC</td></tr>
<tr><td>Assert 2</td><td>0xDD
0xEE
0xxFF
0x00</td><td>0
1
2
3</td><td>10
11
11
01</td><td colspan="2">0xDDEEFF00</td></tr>
<tr><td>Assert 3</td><td>0x1111
0x9999</td><td>0
1</td><td>10
10</td><td colspan="2">0x11119999</td></tr>
</table>

Total ada 9 Assert untuk PDU Dekoder. Masing-masing Fungsi mempunyai 3 Assert. NULL pada Test errorCheck berarti bahwa PDU yang diperiksa Error. Jika tidak NULL, atau tidak Error, maka PDU akan bernilai sama.

TABLE IV berisi keterangan mengenai hasil yang diberikan Unit Test. Jumlah Suits adalah 2 (baris pertama) yaitu PDU Enkoder dan PDU Dekoder. Jumlah Test adalah 7 (baris kedua) Sesuai dengan jumlah Fungsi (lihat TABEL 1). Jumlah Assert adalah 21 karena masing-masing Test berisi 3 Assert. Karena semua Fungsi tidak ada yang Failed dan semua sudah Passed berarti masing-masing Fungsi memberikan Output dari Input sesuai dengan Requirement. Sehingga Fungsi-Fungsi ini dapat digabungkan untuk melalui Fungsional Test.

TABEL IV
HASIL EKSEKUSI UNIT TEST

	Ran	Passed	Failed
Suits	2	2	0
Tests	7	7	0
Asserts	21	21	0

Fungsional Test dilakukan agar diketahui, secara keseluruhan, apakah PDU Enkoder dan PDU Dekoder sudah dapat menjalankan Fungsinya. Skenario pengujian Functional Test adalah memasang PDU Enkoder pada Tx dan PDU Dekoder pada Rx. Karena Fungsi PDU Dekoder adalah membentuk data yang serupa dengan data yang dikirim, maka Pengujian ini dianggap berhasil jika hal tersebut terjadi.

TABEL V memberikan hasil pengujian pada Functional Test. Kolom sebelah kiri memberi keterangan mengenai isi dari Tx SDU Buffer sebelum PDU Enkoder dijalankan. Kolom tengah merupakan PDU-PDU yang terbentuk. Kolom kanan memberi keterangan mengenai isi dari Rx SDU Buffer sesudah PDU Dekoder dijalankan.

TABEL V
HASIL FUNCTIONAL TEST

Input	PDU	Output
CID: 0x4444 SDU: 0x11 0x44444444 0x5555555555 0x666666666666 0x77777777777777	0x824013444403000311000644444444 0x824014444415000755555555558800366 0x824014444415480766666666668800377 0x8240124444BE4808777777777777	CID: 0x4444 SDU: 0x11 0x44444444 0x5555555555 0x666666666666 0x77777777777777
CID: 0xBBBB SDU: 0x44444444 0xCCCCCCCCCCCCCCCCCCCCCCCC	0x80000ABBBBE044444444 0x84000ABBBB6F20CCCCCCCCCCCCCCCCCC 0x840009BBBBD248CCCCCC	CID: 0xBBBB SDU: 0x44444444 0xCCCCCCCCCCCCCCCCCCCCCCCC

PDU dengan CID 0x4444 diberi pengaturan agar dapat melakukan Packing dan melalui CRC Calculation dan Bandwidth Allocation = 20 Byte. Sedangkan CID 0xBBBB diberi pengaturan agar tidak dapat melakukan Packing dan tidak melalui CRC Calculation dan Bandwidth Allocation = 10 Byte.

Pada CID 0x4444, dapat terlihat PDU pertama berisi SDU pertama dan SDU kedua. Di PDU kedua pun tampak SDU kedua, ini menandakan terjadinya Fragmentasi. Ini terjadi juga pada PDU-PDU berikutnya. Di Rx SDU Buffer semua SDU kembali seperti semula.

Pada CID 0xBBBB, dapat terlihat bahwa SDU kedua terdapat pada PDU kedua dan PDU ketiga. Ini menandakan terjadinya Fragmentasi. Di Rx SDU Buffer semua SDU kembali seperti semula. Ini menandakan PDU Enkoder dan PDU Dekoder dapat membentuk data dan mengembalikan data seperti semula.

V. KESIMPULAN

PDU Enkoder dan Dekoder diimplementasikan dalam bentuk software menggunakan bahasa pemrograman C. Total terdapat 7 Fungsi yaitu Packing, Fragmentation, Enkode MAC Header, dan CRC Calculation untuk PDU Enkoder dan Error Check, Unpack, dan Reassembly untuk PDU Dekoder. Semua Fungsi sudah melewati Unit Test.

Secara keseluruhan PDU Enkoder dan PDU Dekoder sudah melewati Functional Test yaitu mengirim data melalui PDU Enkoder dan menerima data melalui PDU Dekoder. Hasil dari pengujian ini adalah Data yang dikirim sama dengan data yang diterima.

Unit Test digunakan untuk memeriksa apakah Fungsi-Fungsi yang dibuat sudah sesuai Standar IEEE 802.16-2012 atau belum. Functional Test digunakan untuk memeriksa apakah PDU Enkoder dan PDU Dekoder dapat berkomunikasi atau tidak. Karena PDU Enkoder dan PDU Dekoder sudah melewati kedua test diambil kesimpulan bahwa kedua program ini dapat dijalankan di atas sebuah perangkat WiMAX (yang diasumsikan diimplementasikan dengan SoC).

Untuk penelitian selanjutnya disarankan untuk melakukan pengujian performansi pada PDU Enkoder dan Dekoder yang telah dijalankan di atas perangkat WiMAX. Selain itu, disarankan untuk melakukan pengembangan dan analisa terhadap modul-modul WiMAX lain seperti Scheduler, ARQ, Enkripsi, PHS, dan lain-lain.

UCAPAN TERIMA KASIH

Terima kasih kepada Allah SWT. Juga kepada Pak Maman dan Pak Adi selaku Dekan dan Wakil Dekan I Fakultas Informatika atas bimbingannya. Terima kasih kepada Mbak Widi yang telah membantu dalam Revisi. Terima kasih kepada Pak Arif Sasongko yang menjadi Pembimbing pada saat penelitian ini berupa Tugas Akhir Sarjana. Terima kasih kepada Pak Andrian sebagai Ketua KK Telematika. Terima kasih kepada Istri saya Ratu Nadia Febrina.

REFERENSI

[1] T. Adiono, R. Ferdian, N. Ahmadi, F. Dawani and I. Abdurrahman, Flexible data sharing architecture of WiMAX heterogeneous Multiprocessor System on Chip, 2015 ISOCC, Gyungju, 2015, pp. 131-132.
[2] IEEE Standards for Local and Metropolitant Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access, 2012
[3] S. Madanapalli, et al.,. Transmission of IPv4 Packets over the IP Convergence Sublayer of IEEE 802.16, RFC 5948, August 2010
[4] A. S. Panda and G. L. Kumar, Comparison of serial data-input CRC and parallel data-input CRC design for CRC-8 ATM HEC employing MLFSR, ICECS, 2014 International Conference on, Coimbatore, 2014, pp. 1-4.
[5] R. Cuellar (2012) Agile Software Development, in The Next Wave of Technologies: Opportunities from Chaos (ed P. Simon), John Wiley & Sons, Inc., Hoboken, NJ, USA
[6] Sujit, V. V. N., and J. Lakshmi Narayana. Analysis of Efficient CRC Implementation Configurations. International Journal of Engineering In Advanced Research Science and Technology (2016): pp. 7880-88
[7] Jamro, Marcin. POU-oriented Unit Testing of IEC 61131-3 Control Software. Industrial Informatics, IEEE Transactions on 11.5 (2015): 1119-1129.
[8] Romano, Simone, et al. Results from an ethnographically-informed study in the context of test-driven development. No. e1864v1. PeerJ Preprints, 2016.