BAB 5 IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi beberapa jenis perantara berbasis DTN dengan mengacu pada bab analisis kebutuhan dan perancangan. Pada penelitian ini, implementasi meliputi: konfigurasi IBR-DTN *service*, implementasi perantara statis, implementasi perantara bergerak tanpa kemampuan unggah, perantara bergerak dengan kemampuan unggah dan pengunaan fitur pada pusat data.

5.1 Implementasi Perantara

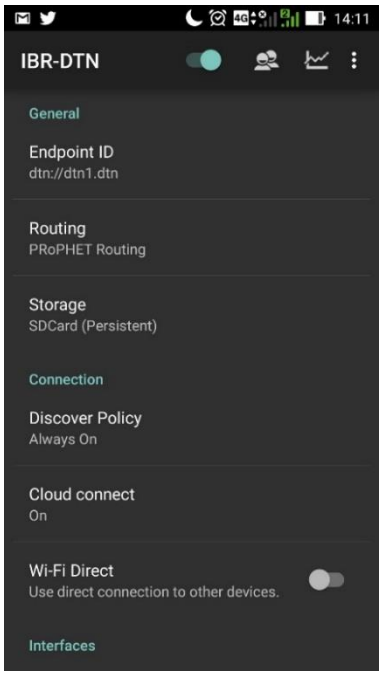
Perantara pada dasarnya merupakan perangkat *mobile* berbasis sistem operasi Android yang terpasang perangkat lunak IBR-DTN. Perangkat IBR-DTN terdiri dari 2 perangkat lunak, yaitu IBR-DTN *service* dan IBR-DTN *client*. Namun, pada perantara bergerak tanpa kemampuan unggah hanya terpasang IBR-DTN *service* saja.

5.1.1 Antarmuka IBR-DTN *Service*

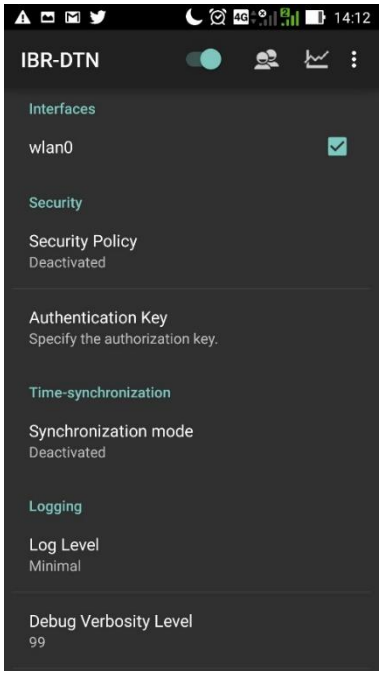
IBR-DTN *service* merupakan perangkat lunak yang bertugas mengatur lalu lintas pesan, pengalamatan pesan, protokol *routing* dan *interface* komunikasi. IBR-DTN *service* dapat diunduh pada "https://play.google.com/store/apps/details?id=de.tubs.ibr.dtn".

5.1.1.1 Antarmuka Konfigurasi

Pada IBR-DTN *service*, *user* dapat mengganti beberapa konfigurasi komunikasi melalui antarmuka pada Gambar 5.1 dan 5.2 berikut:



Gambar 5.1 IBR-DTN *service* 1



Gambar 5.2 IBR-DTN *service* 2

Fungsi dari masing-masing konfigurasi pada antarmuka seperti terlihat pada Tabel 5.1 berikut:

Tabel 5.1 Konfigurasi Komunikasi IBR-DTN Service

No.	Fitur
	Group General
1	Mengganti *endpoint ID* sebagai identitas/alamat dari IBR-DTN *client*
2	Mengganti *routing* sebagai routing protocol yang digunakan dalam berkomunikasi menggunakan mekanisme DTN. Pilihan yang tersedia adalah *disabled, direct-delivery, flooding, epidemic routing, prophet routing.*
3	Mengganti *storage* sebagai tempat penyimpanan pesan yang diterima maupun dikirim. Pilihan yang tersedia adalah SDCard (*Persistent*), SDCard *(Non-Persistent),* dan memori.
	Group Connection
4	Mengganti *discover policy* sebagai kebijakan dalam melakukan penemuan tetangga. Pilihan yang tersedia adalah *off, smart, dan on.*
5	Mengganti *cloud connect* sebagai kebijakan dalam terhubung ke *cloud* DTN. Pilihan yang tersedia adalah *off, WiFi only, dan on.*
6	Menentukan pilihan apakah koneksi dengan WiFi *direct* dizinkan atau tidak.

Group Interface	
7	Menampilkan daftar *interface* komunikasi yang tersedia dan menentukan apakah *interface* tersebut diaktifkan atau tidak

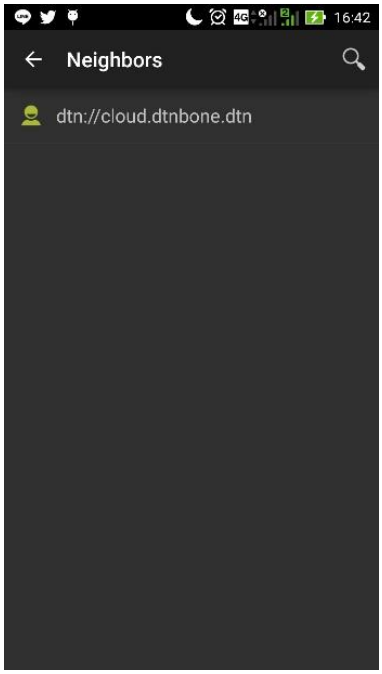
Berikut beberapa konfigurasi yang harus dilakukan pada IBR-DTN *service* agar IBR-DTN *client* dapat berfungsi seperti yang diinginkan:

1. Menyesuaikan endpoint id yang merupakan alamat dari perangkat dalam berkomunikasi menggunakan DTN dengan alamat yang diinginkan.
2. Memilih prophet routing karena *routing protocol* yang digunakan pada penelitian ini adalah prophet.
3. Menonaktifkan WiFi *direct* untuk pilihan koneksi karena pada penelitian ini tidak menggunakan WiFi *direct*. Pada penelitian ini, komunikasi antar-perangkat menggunakan WiFi melalui *access point* karena cara tersebut menghasilkan jalur komunikasi yang lebih stabil dibandingkan dengan WiFi *direct* yang sering mengalami putus koneksi.
4. Mengaktifkan *interface* komunikasi wlan0.

5.1.1.2 Antarmuka Monitoring

Selain digunakan untuk mengganti konfigurasi terkait komunikasi DTN, IBR-DTN *service* juga memiliki fitur untuk mengawasi beberapa kondisi komunikasi yang akan dipakai pada penelitian ini. Fitur-fitur tersebut adalah sebagai berikut:

1. Menampilkan daftar node DTN yang sedang terhubung

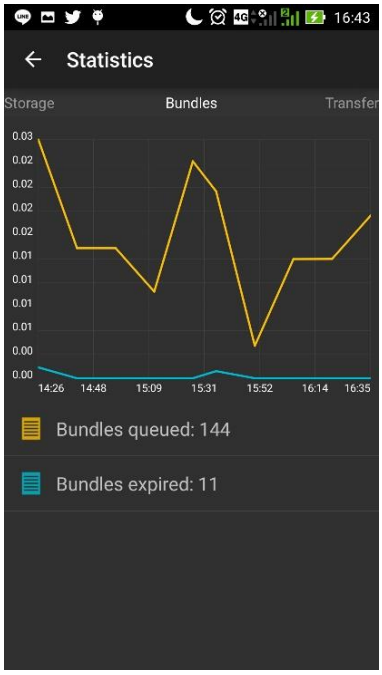


Gambar 5.3 Halaman *Neighbors* pada IBR-DTN *Service*

2. Menampilkan besar *storage* yang terpakai dalam bertukar pesan

Gambar 5.4 Halaman *Statistics-Storage* pada IBR-DTN *Service*

3. Menampilkan jumlah *bundle* yang diantrekan untuk dikirim dan *bundle* yang telah kadaluwarsa dan dikeluarkan dari antrean

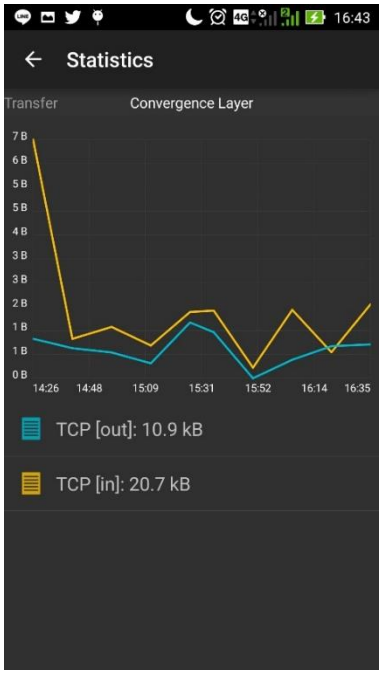


Gambar 5.5 Halaman *Statistics-Bundles* pada IBR-DTN *Service*

4. Menampilkan jumlah *transfer* yang berhasil, *transfer* yang diantrekan kembali dan *transfer* yang digugurkan

Gambar 5.6 Halaman *Statistics-Transfer* pada IBR-DTN *Service*

5. Menampilkan jumlah transaksi TCP keluar dan masuk



Gambar 5.7 Halaman *Statistics-Convergence Layer* pada IBR-DTN *Service*

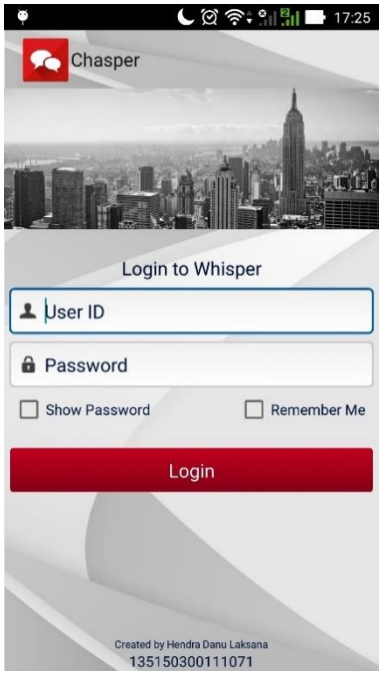
5.1.2 Antarmuka IBR-DTN Client Chasper

IBR-DTN client merupakan perangkat lunak yang bertugas mendefinisikan pesan, alamat tujuan pesan dan proses-proses lain diluar kebutuhan komunikasi,

seperti: menyimpan pesan pada *database* dan menampilkan pesan yang dikirim maupun yang diterima. IBR-DTN *client* adalah perangkat lunak yang dirancang pada penelitian ini dan selanjutnya disebut sebagai Chasper. Chasper merupakan IBR-DTN *client* yang dikembangkan dengan dasar IBR-DTN Whisper yang merupakan perangkat lunak untuk *chatting*. Pengembangan yang dilakukan adalah pada penambahan fitur-fitur mekanisme komunikasi yang kemudian dibahas lebih lanjut pada masing-masing subbab implementasi perantara.

Walaupun ada 2 jenis perantara yang menggunakan IBR-DTN *client*, namun keduanya menggunakan IBR-DTN *client* yang sama yaitu Chasper. Berikut adalah *user interface* dari perangkat lunak Chasper:

1. Halaman *Login* Chasper



Gambar 5.8 Halaman *Login* Chasper

Halaman ini terdiri:

a. Edit text *username* yang digunakan untuk menginputkan *username*

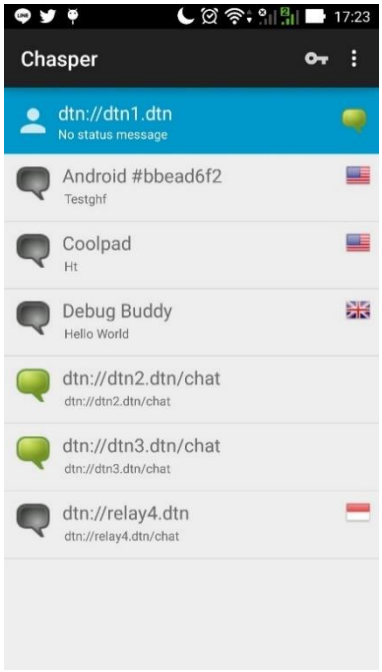
b. Edit text *password* yang digunakan untuk menginputkan *password*

c. Check box *show password* yang digunakan untuk mengaktifkan fungsi *show password* yang dapat mengubah bentuk text pada edit text *password* dari simbol menjadi alfanumerik biasa

d. Check box *remember me* yang digunakan untuk mengaktifkan fungsi *remember me* yang memungkinkan perangkat lunak dapat menyimpan *username* dan *password*

e. Button *login* yang digunakan untuk melakukan perintah *login* menggunakan *username* dan *password* yang tekah diinputkan

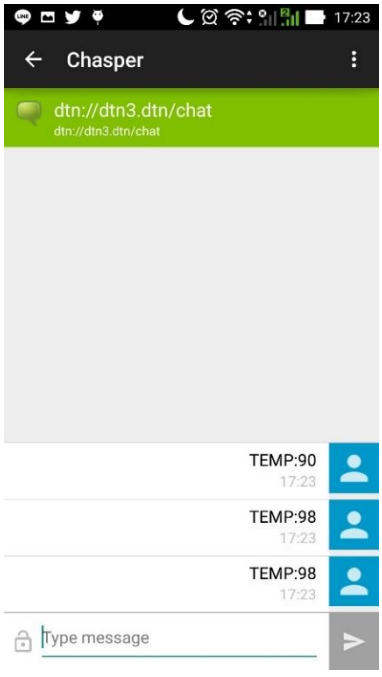
2. Halaman Tujuan Pesan



Gambar 5.9 Halaman Tujuan Pesan

Halaman ini terdiri *list view* yang menampilkan daftar tujuan pengiriman pesan.

3. Halaman Daftar Pesan

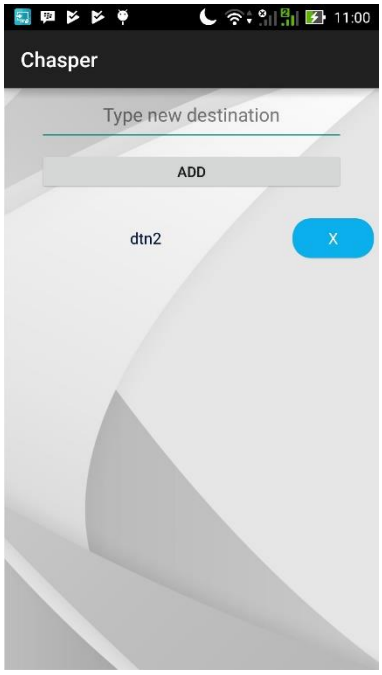


Gambar 5.10 Halaman Daftar Pesan

Halaman ini terdiri *list view* yang menampilkan daftar pesan yang dikirimkan kepada suatu tujuan pengiriman pesan.

5.1.3 Implementasi Komunikasi Perantara Statis

Perantara statis merupakan sebuah perantara yang ditempatkan di lingkungan jaringan sensor. Perantara jenis ini tidak berpindah tempat, maka dari itu disebut perantara statis. Perangkat lunak Chasper dapat digunakan sebagai perantara statis dengan cara melakukan *login* menggunakan *bypass username* "mqtt". Ketika *login* menggunakan *username* tersebut, maka *password* tidak diperdulikan seperti yang telah dibahas pada bab perancangan sebelumnya. Jika login sebagai perantara statis, maka akan muncul pilihan menu "Add Destination" pada *option menu.* Menu tersebut digunakan untuk menambah atau mengurangi daftar perantara bergerak dengan kemampuan unggah sebagai tujuan pengiriman data sensor. Antarmuka dari halaman "Add Destination" dapat dilihat pada Gambar 5.11:



Gambar 5.11 Halaman *Add Destination*

Halaman antarmuka ini terdiri dari:

1. *Edit text* yang digunakan untuk menginputkan alamat perantara bergerak dengan kemampuan unggah
2. Button *add* yang digunakan untuk menambah alamat yang telah diinputkan pada *edit text* ke *list view* alamat
3. *List view* alamat digunakan untuk menampilkan daftar alamat perantara bergerak dengan kemampuan unggah sebagai tujuan pengiriman data sensor
4. Button *delete* yang digunakan untuk menghapus item pada *list view* alamat

Pada bab perancangan juga telah dijelaskan bahwa perantara statis mempunyai fungsi menjembatani pengiriman data sensor dari *sensor node* ke perantara bergerak. Agar dapat melakukan fungsi tersebut, perantara statis harus memiliki 2 peran, yaitu:

5.1.3.2 Perantara Statis Sebagai Subscriber

Jaringan sensor pada penelitian ini menggunakan protokol MQTT dalam berkomunikasi antar *node*-nya. Pada sudut pandang ini, perantara statis berperan sebagai *subscriber* yang menerima data sensor dari *sensor node* melalui broker. Oleh karena itu agar perantara statis dapat menerima data sensor dari *sensor node*, maka perantara statis harus melakukan *subscribe* dengan topik "/test". Beberapa potongan kode pada Tabel 5.2 adalah kode yang digunakan oleh perantara statis agar dapat menjadi *subscriber* pada jaringan sensor:

Tabel 5.2 Potongan Kode Perantara Statis Sebagai *Subscriber*

No. Blok	Kode

1	`@Override`
	`public void onCreate(@Nullable Bundle savedInstanceState) {`
	`super.onCreate(savedInstanceState);`

2	`if(GeneralData.getUserId().equalsIgnoreCase(GlobalData.mqttUser)){`
3	`PahoMQTT mqttClient = new PahoMQTT();` `mqttClient.initPahoMQTT(getContext());`

4	`mqttClient.setOnActionListener(new IMqttMessageListener() {`
	`@Override`
	`public void messageArrived(String topic, MqttMessage message) throws Exception {`
	`buffer = new String(message.getPayload());`

Berikut penjelasan dari beberapa blok kode pada Tabel 5.2:

Blok 1. Merupakan kode yang menunjukkan bahwa blok kode dibawahnya dijalankan pada fungsi onCreate. Fungsi onCreate dijalankan ketika fragment pertama kali terbentuk.

Blok 2. Merupakan kode yang digunakan untuk melakukan pengecekan apakah *user* yang sedang *login* ada *bypass user* atau tidak.

Blok 3. Merupakan kode yang digunakan untuk membuat MQTT client object, menginisiasi koneksi ke broker dan melakukan *subscribe* ke topik "/test".

Blok 4. Merupakan kode yang digunakan untuk memasang onActionListener pada object yang telah dibuat sebelumnya. Listener secara otomatis memicu callback method messageArrived ketika ada pesan masuk.

5.1.3.3 Perantara Statis Sebagai Node DTN

Setalah menerima data sensor dari *sensor node*, maka selanjutnya data sensor tersebut dikirim ke perantara bergerak dengan kemampuan unggah agar bisa di

unggah ke pusat data. Perantara bergerak dengan kemampuan unggah yang menjadi tujuan pengiriman adalah perantara yang telah didaftarkan alamatnya pada halaman "Add Destination" yang dapat dilihat pada Gambar 5.11. Pada sudut pandang ini, perantara statis berperan sebagai node DTN. Beberapa potongan kode pada Tabel 5.3 adalah kode yang digunakan oleh perantara statis agar dapat menjadi node DTN yang bertugas meneruskan data sensor ke perantara bergerak:

Tabel 5.3 Potongan Kode Perantara Statis Sebagai Node DTN

No. Blok	Kode
1	`if(GeneralData.getBuddyEndpointMap() != null && GeneralData.getBuddyEndpointMap().size() != 0){`
2	`for (HashMap.Entry<String, Long> entry : GeneralData.getBuddyEndpointMap().entrySet()) { Long buddyId = entry.getValue(); forwardMessage(buffer, buddyId); } } } });` `} }` `public void forwardMessage(String message, Long buddyId){ if(!LoginActivity.wiFiConnection.isConnected()){ LoginActivity.wiFiConnection.connect(); } final Intent intent = new Intent(getActivity(), ChatService.class); intent.setAction(ChatService.ACTION_SEND_MESSAGE); intent.putExtra(ChatService.EXTRA_BUDDY_ID, buddyId); intent.putExtra(ChatService.EXTRA_TEXT_BODY, message); getActivity().startService(intent); }`

Berikut penjelasan dari beberapa blok kode pada Tabel 5.3:

Blok 1. Merupakan kode yang digunakan untuk melakukan pengecekan apakah pada perantara statis ada tujuan pengiriman data sensor yang didefinisikan.

Blok 2. Merupakan kode yang digunakan untuk meneruskan data sensor menggunakan mekanisme DTN dari *sensor node* kepada semua perantara bergerak dengan kemampuan unggah yang telah didaftarkan pada perantara statis sebagai tujuan pengiriman. Pengiriman pesan dilakukan menggunakan fungsi pengiriman pesan milik IBR-DTN *client* Whisper.

5.1.4 Implementasi Komunikasi Perantara Bergerak tanpa Kemampuan Unggah

Perantara bergerak tanpa kemampuan unggah adalah perantara yang berpindah-pindah tempat, tetapi tidak memiliki kemampuan unggah. Ketika sebuah perangkat akan dijadikan sebagai perantara bergerak tanpa kemampuan unggah, maka perangkat tersebut hanya memerlukan IBR-DTN *service* saja. IBR-DTN *client* tidak diperlukan karena tugas dari perantara jenis ini hanyalah meneruskan data sensor hingga sampai ke perantara bergerak dengan kemampuan unggah. Seperti yang telah dijelaskan sebelumnya, tugas semacam itu telah selesai ditangani oleh IBR-DTN *service*.

5.1.5 Implementasi Komunikasi Perantara Bergerak dengan Kemampuan Unggah

Perantara bergerak dengan kemampuan unggah merupakan sebuah perantara yang berpindah-pindah tempat dan memiliki kemampuan unggah. Perangkat lunak Chasper dapat digunakan sebagai perantara bergerak dengan kemampuan unggah dengan cara melakukan *login* menggunakan *user* yang telah didaftarkan pada pusat data, yaitu *username* “basukicaya” dan *password* “rahasia”.

Pada bab perancangan juga telah dijelaskan bahwa perantara bergerak dengan kemampuan unggah mempunyai fungsi mengunggah data sensor ke pusat data. Agar dapat melakukan fungsi tersebut, perantara statis harus memiliki 2 peran, yaitu:

5.1.5.1 Perantara Bergerak dengan Kemampuan Unggah Sebagai Node DTN

Sebelum melakukan unggah data ke pusat data, perantara begerak dengan kemampuan unggah menerima data sensor dari perantara statis atau perantara bergerak lainnya. Pada sudut pandang ini, perantara bergerak dengan kemampuan unggah berperan sebagai node DTN. Beberapa potongan kode pada Tabel 5.4 adalah kode yang digunakan oleh perantara bergerak dengan kemampuan unggah agar dapat menjadi node DTN:

Tabel 5.4 Potongan kode Perantara Bergerak dengan Kemampuan Unggah Sebagai Node DTN

No. Blok	Kode
1	`private DataHandler _data_handler = new DataHandler() {`
2	`public void endBlock() {`

3	`if (stream != null) {`
	`String msg = new String(stream.toByteArray());`
	`stream = null;`
	`if`
	`(current.getDestination().equals(PRESENCE_GROUP_EID)`
	`{`

	eventNewPresence(current.getSource(),
	current.getTimestamp().getDate(), msg, flags);
	}
	else
	{
	eventNewMessage(current.getSource(),
	current.getTimestamp().getDate(), msg, flags);
	}
	}
	}
	}
	private void eventNewMessage(SingletonEndpoint
	source, Date created, String payload, Long flags)
	{

4	Long msgId =
	getRoster().createMessage(source.toString(), created,
	new Date(), true, payload, flags);
	Message msg = getRoster().getMessage(msgId);
	Buddy b = getRoster().getBuddy(msg.getBuddyId());
	createNotification(b, msg);

5	SensorMessage sensorMessage = new SensorMessage();
	sensorMessage.setIs_sent("false");
	String keyword;
	String value;
	int delimiter = payload.indexOf(':');
	if (delimiter == -1) {
	keyword = "";
	value = payload;
	}else{
	keyword = payload.substring(0, delimiter);
	value = payload.substring(delimiter + 1,
	payload.length()).trim();
	}
	sensorMessage.setMessage(value);
	sensorMessage.setSensor_type(keyword);
	sensorMessage.setSource(source.toString());
	String uuid = Tool.getUUID();
	sensorMessage.setUuid_sensor_message(uuid);
	SensorMessageDataAccess.addOrReplace(getApplicationCo
	ntext(), sensorMessage);
	}

Berikut penjelasan dari beberapa blok kode pada Tabel 5.4:

Blok 1. Merupakan kode yang menujukkan bahwa blok kode dibawahnya merupakan bagian dari DataHandler. DataHandler akan dijalankan ketika ada pesan masuk. Cara penerimaan pesan ini merupakan cara penerimaan pesan milik IBR-DTN *client* Whisper.

Blok 2. Merupakan kode yang menujukkan bahwa blok kode dibawahnya merupakan bagian dari fungsi endBlock. Fungsi endBlock akan dijalankan ketika blok pesan yang masuk selesai diproses.

Blok 3. Merupakan kode yang digunakan untuk melakukan pengecekan apakah pesan yang diterima adalah data sensor atau pemberitahuan kehadiran node DTN lain. Jika pesan yang diterima adalah data sensor, maka fungsi eventNewMessage akan dijalankan.

Blok 4. Merupakan kode yang digunakan untuk membuat notifikasi yang memberitahukan bahwa ada pesan masuk kepada *user*.

Blok 5. Merupakan kode yang digunakan untuk menyimpan data sensor yang diteriman ke *database*.

5.1.5.2 Perantara Bergerak dengan Kemampuan Unggah Sebagai HTTP Client

Setelah menerima data sensor dari perantara statis atau perantara bergerak lainnya, selanjutnya perantara bergerak dengan kemampuan unggah akan mengunggah data tersebut ke pusat data melalui jaringan seluler. Pada sudut pandang ini, perantara bergerak dengan kemampuan unggah berperan sebagai HTPP *client*. Beberapa potongan kode pada Tabel 5.5 adalah kode yang digunakan oleh perantara bergerak dengan kemampuan unggah agar dapat menjadi node DTN:

Tabel 5.5 Potongan Kode Perantara Bergerak dengan Kemampuan Unggah Sebagai HTTP Client

No. Blok	Kode

1	`private void uploadData(){`
	`try{`
	`if(LoginActivity.wiFiConnection.isConnected()){`
	`LoginActivity.wiFiConnection.disconnect();`
	`Thread.sleep(1000);`
	`}`

2	`if(Tool.isInternetConnected(getApplicationContext()))` `{`

3	`List<SensorMessage> sensorMessageList =`
	`SensorMessageDataAccess.getAllIsNotSent(getApplicatio`
	`nContext());`
	`List<SubMessageRequestBean> subBeanList = new`
	`ArrayList<SubMessageRequestBean>();`
	`if(sensorMessageList != null){`
	`for(SensorMessage sensorMessage:sensorMessageList){`
	`if(Arrays.asList(GlobalData.sensorList).contains(sens`
	`orMessage.getSensor_type())){`
	`SubMessageRequestBean subBean = new`
	`SubMessageRequestBean();`
	`subBean.setSensor(sensorMessage.getSensor_type());`
	`subBean.setData(sensorMessage.getMessage());`

	subBeanList.add(subBean);
	}else{
	SensorMessageDataAccess.delete(getApplicationContext(
	), sensorMessage);
	sensorMessageList.remove(sensorMessage);
	}
	}

4	SubMessageRequest subRequest = new
	SubMessageRequest();
	subRequest.setSubMessageRequestBeanList(subBeanList);
	String jsonRequest = GsonHelper.toJson(subRequest);
	SubMessageResponse subResponse;
	String jsonResponse;
	HttpClient httpClient = new HttpClient();
	jsonResponse = httpClient.postMessage(jsonRequest,
	url+subs, GeneralData.getToken());

5	if(jsonResponse != null){
	if(jsonResponse.equalsIgnoreCase("retryToken")){
	getToken();
	uploadData();
	}else{

6	for(SensorMessage sensorMessage:sensorMessageList){
	sensorMessage.setIs_sent("true");
	}
	SensorMessageDataAccess.addOrReplace(getApplicationCo
	ntext(), sensorMessageList);
	Toast.makeText(getApplicationContext(), "Sending
	Message Success", Toast.LENGTH_LONG).show();
	}
	}else{
	Toast.makeText(getApplicationContext(), "Sending
	Message Error", Toast.LENGTH_LONG).show();
	}
	}
	}

7	if(!LoginActivity.wiFiConnection.isConnected()){
	LoginActivity.wiFiConnection.connect();
	}
	}catch(Exception e){
	e.printStackTrace();
	}
	}

Berikut penjelasan dari beberapa blok kode pada Tabel 5.5:

Blok 1. Merupakan kode yang digunakan untuk menonaktifkan koneksi WiFi sebelum melakukan unggah data karena data seluler tidak aktif ketika koneksi WiFi aktif.

Blok 2. Merupakan kode yang digunakan untuk melakukan pengecekan apakah ada koneksi internet atau tidak.

Blok 3. Merupakan kode yang digunakan untuk melakukan pengecekan format data sensor. Jika format data sensor salah, maka data tersebut akan dihapus. Format data sensor yang benar adalah yang memiliki keyword diantara “TEMP”, “RADIANCE”, atau “HUMIDITY”.

Blok 4. Merupakan kode yang digunakan untuk menyiapkan JSON *request* dan melakukan *request* ke pusat data.

Blok 5. Merupakan kode yang digunakan untuk melakukan pengecekan *response* yang dikirimkan oleh pusat data. Jika *response* menandakan bahwa token telah kadaluwarsa, maka dilakukan *request* ulang token dan unggah ulang data sensor.

Blok 6. Merupakan kode yang digunakan untuk menandai data sensor yang telah berhasil terkirim pada *database*.

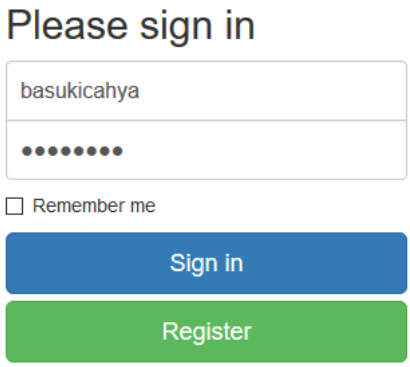
Blok 7. Merupakan kode yang digunakan untuk mengaktifkan WiFi kembali setelah unggah data sensor selesai. Hal ini dilakukan agar perantara bergerak dengan kemampuan unggah dapat menerima data sensor dari perantara statis atau perantara bergerak lain lagi.

5.2 Penggunaan Pusat Data

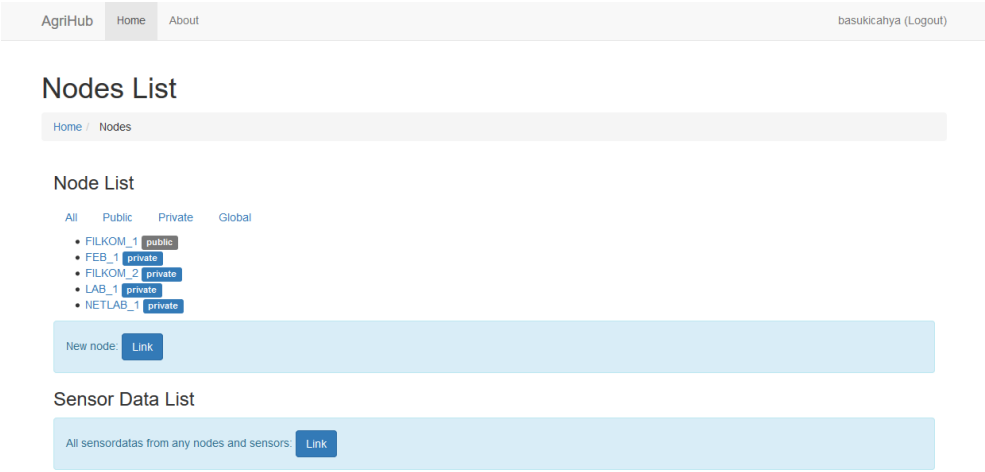
Pusat data yang dipakai dalam penelitian ini adalah pusat data yang dikembangkan oleh Ocki Bagus Pramata yang merupakan seorang mahasiswa Universitas Brawijaya. Pusat data tersebut dapat diunduh pada “https://github.com/OckiFals/cloud-gateway”. Pusat data tersebut terdiri dari 2 komponen, yaitu sebagai webapp dan webservice.

5.2.1 Penggunaan Webapp pada Pusat Data

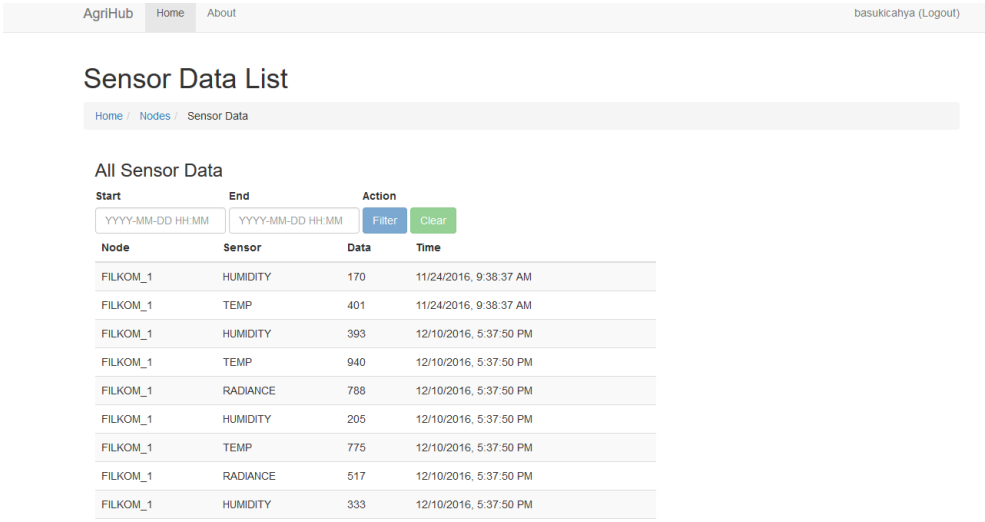
Webapp bertugas menyajikan data pada laman web, sehingga data-data yang ada pada pusat data dapat dibaca oleh *user*. Antarmuka webapp dapat dilihat pada Gambar 5.12, Gambar 5.13, dan Gambar 5.14 berikut:



Gambar 5.12 Halaman *Login* Webapp



Gambar 5.13 Halaman Utama Webapp



Gambar 5.14 Halaman Daftar Data Webapp

Cara mengakses webapp tersebut hingga dapat menampilkan data sensor, seperti yang dijelaskan berikut:

1. Melakukan *login* menggunakan *user* yang telah terdaftar, yaitu *username* "basukicahya" dan *password* "rahasia".
2. Menekan button "Link" pada submenu "Sensor Data List"

5.2.2 Penggunaan Webservice pada Pusat Data

Webservice bertugas sebagai API yang dapat dipergunakan sebagai jalur input-output data dari dan ke perantara bergerak dengan kemampuan unggah yang terhubung ke pusat data. API yang disediakan dapat diakses menggunakan HTTP *request* dan balasan dari pusat data berupa HTTP *response*. Beberapa API yang disediakan dan digunakan pada penelitian ini dapat dilihat pada Tabel 5.6 berikut:

Tabel 5.6 Daftar API Webservice

No.	API		Contoh Konten JSON	Fungsi
	URL	Header		

1	http://agrihub.tujuhlangit.id:8080/node-auth/	Name: -
		Value: -
			{
			"user":"basukicahya",
			"label":"FILKOM_1",
			"secretkey":"rahasia"
			}
				Untuk meminta token

2	http://agrihub.tujuhlangit.id:8080/subscriptions/	Name: Authorization
		Value: Token yang didapat dari server ditambah dengan prefix “JWT<space>”
			{
			"publish": [
			{
			"sensor": "HUMIDITY",
			"data": "145"
			},
			{
			"sensor": "TEMP",
			"data": "456"
			},
			{
			"sensor": "RADIANCE",
			"data": "356"
			}
			]
			}
				Untuk mengunggah data