BAB 5 IMPLEMENTASI

5.1 Implementasi *agent*

Pada implementasi *Agent* disini memakai bahasa pemrograman Python melalui juga library *framework Twisted dan library psutil* sebagai pengambilan data proses *monitoring*. Implementasi agent disesuaikan dengan rancangan yang telah dibuat.

5.1.1 Proses pengambilan data pada *agent*

Data pada tabel *resource* yang berisi *cpu usage, memory used, memory available* dan *swap* direpresentasikan dengan komponen yang diambil dari *library* psutil. Jenis data dan komponen dapat dilihat pada Tabel 5.1.

Tabel 5.1 Komponen resource dari library psutil

Jenis Data Resource	Nama Komponen pada library psutil
CPU Usage	psutil.cpu_percent
Memory Used	psutil.virtual_memory().used
Memory Available	psutil.virtual_memory().available
Swap	psutil.virtual_lmemory().free

Source code 5.1 Method getResource

```
1   def getResource():
2       _type = 1
3       cpu = psutil.cpu_percent(interval=1)
4       memory = psutil.virtual_memory().available
5       usedmem = psutil.virtual_memory().used
6       swap = psutil.swap_memory().free
7       res = {
8           'type': _type,
9           'mac': getmac(),
10          'cpu': cpu,
11          'memory': memory,
12          'usedmem': usedmem,
13          'swap': swap
14      }
15      result = json.dumps(res)
16      return result
17
```

Pada Source code 5.1 dapat dilihat bahwa method getResource digunakan untuk mendapatkan komponen *cpu, memory used, memory available*, dan *swap*. Komponen tersebut kemudian dibentuk dengan struktur data *dictionary*. Dari Struktur data *dictionary* itu lalu diubah ke dalam format *JSON untuk proses pertukaran data*.

Data pada tabel network yang berisi *byte sent, byte receive, packet sent, packet receive* adalah komponen yang diambil dari library psutil. Dari *library psutil* terdapat beberapa perintah yang dapat dipakai untuk mengambil data dalam katergori *network*. Jenis data dan komponen tabel *Network* dapat dilihat pada Tabel 5.2.

Tabel 5.2 Komponen Network dari library psutil

Jenis Data Network	Nama Komponen pada library psutil
Byte Sent	psutil.net_io_counters().bytes_sent
Byte Receive	psutil.net_io_counters().bytes_recv
Packet Sent	psutil.net_io_counters().packets_sent
Packet Receive	psutil.net_io_counters().packets_recv

Source code 5.2 Method Get Network

```
1   def getNetwork():
2       _type = 2
3       bytesent = psutil.net_io_counters().bytes_sent
4       byterecv = psutil.net_io_counters().bytes_recv
5       pktsent = psutil.net_io_counters().packets_sent
6       pktrcv = psutil.net_io_counters().packets_recv
7       net = {
8           'type': _type,
9           'mac': getmac(),
10          'bytesent': bytesent,
11          'byterecv': byterecv,
12          'pktsent': pktsent,
13          'pktrcv': pktrcv
14      }
15      result = json.dumps(net)
16    return result
17
```

Pada Source code 5.2 bahwa *method getNetwork* digunakan untuk mendapatkan komponen *byte sent, byte receive, packet sent, packet receive*. Komponen tersebut kemudian dibentuk dengan struktur data *dictionary*. Dari Struktur data *dictionary* itu lalu diubah ke dalam format *JSON untuk proses pertukaran data*.

5.1.2 *Agent* menerima perintah dari *manager*

Agent dapat menerima perintah *monitoring* yang dikirimkan oleh *Manager* dan memilahnya berdasarkan tipe *monitoring*.

Source code 5.3 Method Menerima Data Perintah *Monitoring*

```
1   def datagramReceived(self, datagram, (host, port)):
2           command = json.loads(datagram)
3           print 'Datagram received: ', repr(command)
4           if None is not command:
5               starttime = datetime.strptime(command[4], "%Y/%m/%d
6   %H:%M:%S.%f")
```

```
7               endtime = datetime.strptime(command[5], "%Y/%m/%d
8   %H:%M:%S.%f")
9               now = datetime.now()
10
11              if command[2] == 1:
12                  callback = getResource()
13              elif command[2] == 2:
14                  callback = getResource()
15              elif command[2] == 3:
16                  callback = getResource()
17              elif command[2] == 4:
18                  callback = getResource()
19              elif command[2] == 10:
20                  callback = getResource()
21              elif command[2] == 5:
22                  callback = getNetwork()
23              elif command[2] == 6:
24                  callback = getNetwork()
25              elif command[2] == 11:
26                  callback = getNetwork()
27              elif command[2] == 7:
28                  callback = ping()
29              elif command[2] == 8:
30                   callback = getDisk()
31
```

Pada Source code 5.3 dapat dilihat bahwa *method datagramReceived* digunakan untuk menerima data berupa perintah *monitoring*. Pada baris 2, data yang diterima berupa JSON, setelah itu berdasarkan data yang dikirim pada *command*[2] berfungsi untuk menentukan kategori OID yang dipilih sehingga akan masuk dalam pemilihan kondisi (baris 11 – 30) . Setelah itu pada baris 5-9 bagian ini masuk penentuan waktu yang yang diterima nilai dari *command*[4] dan *command*[5].

5.1.3 *Agent* mengirim hasil *monitoring* pada *manager*

Setelah *agent* menerima data perintah *monitoring*, *agent* mengirimkan data hasil *monitoring* sesuai dengan kategori OID pada perintah *monitoring*.

Source code 5.4 Pengiriman Data Hasil *Monitoring*

```
1     def transporthandler(self, args):
2          if self.ACTION['SENDING'] == args[0]:
3              print 'sending...'
4          elif self.ACTION['WAITING'] == args[0]:
5              print 'waitting command'
6           self.transport.write(args[1])
7
```

Pada Source code 5.4 dapat dilihat bahwa, pengiriman data hasil *monitoring* sesuai dengan yang kategori OID yang diterima, jika yang diminta adalah data berupa *resource* maka data yang dikirim atau Jika data yang diminta adalah data berupa *network* maka data yang dikirim.

5.2 Implementasi *manager*

Implementasi *manager* menggunakan bahasa pemrograman Python dengan menggunakan *framework Twisted*. *Manager* yang dikembangkan harus memiliki fungsi-fungsi yang sudah dijabarkan di bagian perancangan *manager*.

5.2.1 Proses pengiriman data perintah

Manager mengirim perintah *monitoring* berdasarkan parameter yang diatur pengguna yang disimpan di dalam *database*.

Source code 5.5 Pengiriman Perintah *Monitoring*

```
1  def handleMessage(self, host):
2          db = sqlite3.connect("monitor.db", timeout=20)
3          cursor = db.cursor()
4          row_monitor = cursor.execute('''SELECT * FROM monitor ORDER BY
5  createdat DESC ''')
6          selection = list(row_monitor.fetchone()) or []
7          db.commit()
8          row_host = cursor.execute('''SELECT phase FROM Host WHERE host = ?
9  ''', (selection[1],))
10         d = row_host.fetchone()[0]
11         print d
12         selection.append(d)
13         command = json.dumps(selection)
14         db.commit()
15         db.close()
           self.transport.write(command, (self.host, self.port))
           print "this " + command + " has been sent "
```

Pada Source code 5.5 dapat dilihat bahwa untuk mengirim perintah *monitoring*, perintah diambil dari satu *row* pada tabel *monitor* yang terdapat pada *database*. Row yang diambil adalah row yang dipilih (baris 7-8). Pada bagian ini juga dilakukan pengecekan apakah host yang akan menjadi tujuan dalam fase aktif atau tidak. Jika sudah dalam keadaan aktif maka data perintah dikirimkan ke *agent*.

5.2.2 *Manager* menerima perangkat yang terhubung

Manager dapat menerima perangkat yang terhubung dan menyimpan perangkat ke dalam *database*.

Source code 5.6 Menerima Perangkat yang Terhubung

```
1  def startProtocol(self):
2          print 'server started'
3          db = sqlite3.connect("monitor.db")
4          cursor = db.cursor()
5          query = cursor.execute('''SELECT * FROM Host''')
6          for host in query.fetchall():
7              cursor.execute(
8                  '''UPDATE host SET status = ?, phase = ? WHERE Host =
9  ?;''',
10                 ('Down', 'stopped', host[2])
11             )
12             db.commit()
           db.close()
```

Pada Source code 5.6 dapat dilihat bahwa bagian ini berfungsi untuk menerima data perangkat yang terhubung, jika perangkat baru pertama kali terhubung maka dilakukan perintah *insert* pada *database*, jika sudah pernah terhubung maka yang dilakukan adalah melakukan update pada salah satu kolom di *row* yaitu status perangkat.

5.2.3 Menerima hasil *monitoring* dan menyimpan ke *database*

Manager dapat menerima hasil *monitoring* kemudian menyimpan ke dalam *database*.

Source code 5.7 Menerima Data dan Menyimpan ke *Database*

```
def datagramReceived(self, data, (host, port)):
        handler = Handler(self.transport, host, port)
        result = json.loads(data)
        args = result.get('type') if result.get('type') else
result.get('hostname')
        d = threads.deferToThread(handler.handleMessage, args)
        db = sqlite3.connect("monitor.db")
        cursor = db.cursor()

        if result.get('type') == 1:
            cursor.execute('''INSERT INTO
resource(host,date,time,cpu,memory_avail,memory_used,swap_free)
            VALUES (?,?,?,?,?,?,?);''', (
                result.get('mac'), str(time.strftime("%d-%m-%Y")),
                str(time.strftime("%H:%M:%S")), result.get('cpu'),
result.get('memory'),
                result.get('usedmem'),
                result.get('swap')))
            print "resource has been inserted to database"
            print "result %r " % result
        elif result.get('type') == 2:
            cursor.execute(
                '''INSERT INTO
network(host,date,time,byte_sent,byte_receive,packet_sent,packet_receive)
                VALUES (?,?,?,?,?,?,?);''', (
                    result.get('mac'),
                    str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                    result.get('bytesent'),
                    result.get('byterecv'),
                    result.get('pktsent'),
                    result.get('pktrcv')
                )
            )
            print "network has been inserted to database"
            print "result %r " % (result)
        elif result.get('type') == 3:
            cursor.execute(
                '''INSERT INTO availability(host,date,time,status)
                VALUES (?,?,?,?);''',
                (result.get('mac'),
                 str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                 result.get('avai'))
            )
            print "this is availability"
            print "result %r " % (result)
            print type(result)
        elif result.get('type') == 4:
            cursor.execute(
                '''INSERT INTO
disk(host,date,time,disk_used,disk_free,read_bytes,write_bytes)
                VALUES (?,?,?,?,?,?,?);''', (
                    result.get('mac'), str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                    result.get('disk_used'), result.get('disk_free'),
                    result.get('read_bytes'), result.get('write_bytes'))
            )
```

```
59              print "disk has been inserted to database"
60              print "result %r " % result
61          elif result.get('type') == 5:
62              print 'agent wait for command'
63          else:
64              print "received from host %r , %s:%d" % (result.get('mac'), ho
65  port)
66              t = (result.get('mac'),)  # s = (result[1],)
67              query = cursor.execute('''SELECT * FROM Host WHERE host=?''',
68              count = len(query.fetchall())
69
70              if count == 0:
71                  cursor.execute(
72                      '''INSERT INTO host(hostname,host,status,phase) VALUES
73  (?,?,?,?);''',
74                      (result['hostname'], result['mac'], result['status'],
75  result['phase'])
76                  )
77                  print "insert"
78              else:
79                  cursor.execute(
80                      '''UPDATE Host SET status = ?, phase = ? WHERE hostnam
81  ?;''',
82                      (result['status'], result['phase'], result['hostname']
83                  )
84                  print "update"
85              print "host has been recorded"
86          db.commit()
87          db.close()
88      d.addCallback(handler.postHandle)
89
```

Pada Source code 5.7 dapat dilihat bahwa bagian ini digunakan untuk menerima data dan kemudian menyimpan data tersebut ke dalam *database*. Pada baris 3, data yang diterima diubah format datanya dari JSON ke dalam *dictionary*. Setelah itu data akan masuk ke dalam kondisi sesuai dengan data yang diterima, jika *result*[0] == 1 maka data *resource* yang diterima kemudian disimpan dalam tabel *resource*, jika r*esult*[0]==2 maka data *network* yang diterima kemudian disimpan dalam tabel *network.*

5.3 Implementasi *database*

Database pada sistem *monitoring* ini menggunakan SQLite sebagai *database engine.* Tabel yang dibuat sama dengan perancangan *database* di bagian 4.8.

Gambar 5.1 Skema Tabel *host*

Seperti pada Gambar 5.1 Tabel host terdapat beberapa kolom seperti pada hostname,host, status, phase. Tabel ini untuk menyimpan perangkat yang

terhubung dan status perangkat. Terdapat kolom *phase* untuk kebutuhan pengiriman perintah *monitoring*.

Seperti pada Gambar 5.2 Tabel *monitor* terdapat beberapa kolom seperti pada *host, monitype, interval, start, end, createdat*. Tabel ini untuk menyimpan perintah *monitoring* yang akan dikirimkan

Gambar 5.2 Skema Tabel monitor

Gambar 5.3 Skema Tabel Network

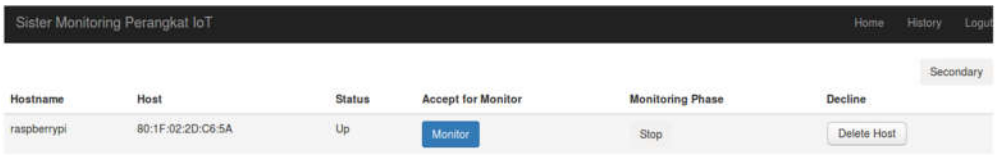
Seperti pada Gambar 5.3, Tabel *network* terdapat beberapa kolom seperti pada *host, date, time, byte_sent, byte_receive, packet_sent, packet_receive*. Tabel ini untuk menyimpan resource berupa *network*.

Gambar 5.4 Skema Tabel Resource

Seperti pada Gambar 5.4, Tabel *resource* terdapat beberapa kolom seperti pada *host, date, time, cpu, memory_avail, memory_used, swap_free*. Tabel ini untuk menyimpan resource berupa *resource*.

5.4 Implementasi aplikasi web

Implementasi Aplikasi Web pada Sistem *Monitoring* menggunakan web *framework Flask*. Aplikasi Web yang dikembangkan harus disesuikan dengan perancangan aplikasi web.



Gambar 5.5 Halaman Utama Sistem *Monitoring*

Pada Gambar 5.5 adalah halaman utama dari Sistem *Monitoring IoT,* pada halaman ini terdapat beberapa fitur pada halaman ini. Yaitu fitur *tampil list device, host, status, monitor, monitoring phase* dan hapus *device* yang tidak ingin diamati.

5.4.1 Fitur tampil list device

Fitur ini dapat menampilkan perangkat *Agent* yang terhubung dengan m*anager*

Source code 5.8 List Device

```
1   @app.route('/index')
2   def index():
3       app.secret_key = os.urandom(12)
4
5       if not session.get('logged_in'):
6           return render_template('login.html')
7       else:
8           app.secret_key = os.urandom(12)
9           rows = models.Host.query.all()
10          host = models.Host.query.options(load_only('host'))
11      return render_template('index.html', title='Home', rows=rows, host=host)
```

Seperti pada *Source code* 5.8, *list device* ditampilkan dengan cara *query* yang dilakukan pada tabel *host* dengan kolom nama host, sehingga akan tampil *list device* yang terhubung.

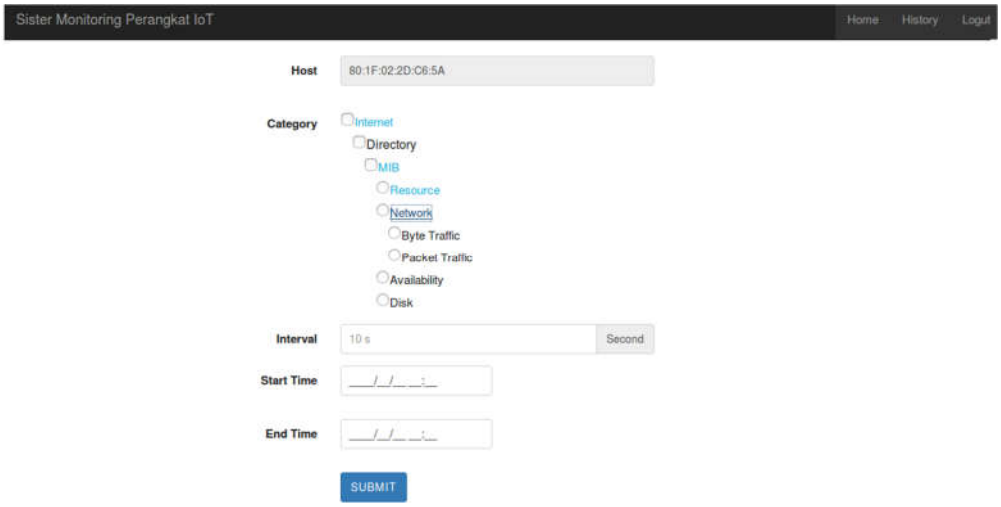
5.4.2 Fitur delete device

Fitur ini berfungsi untuk menghapus suatu *host* dari *database*. Implementasi fitur ini dapat dilihat pada Source code 5.9.

Source code 5.9 Delete Host

```
1   @app.route('/delete_host', methods=['POST'])
2   def delete_host():
3       host = request.form['host']
4       models.Host.query.filter(models.Host.host == host).delete()
5       models.db.session.commit()
6     return redirect('/')
```

Delete Device dilakukan dengan cara mengambil nilai *host* yang dipilih melalui *form* kemudian melakukan *query* seleksi berdasarkan *host* yang dipilih kemudian dihapus. Perangkat yang sudah dihapus tidak bisa diamati, perangkat tersebut bisa masuk dalam daftar kembali ketika melakukan proses terhubung lagi di waktu berikutnya.



Gambar 5.6 Halaman *Add Monitoring*

Pada Gambar 5.6 adalah halaman untuk menambahkan perintah *monitoring* pada Sistem *Monitoring IoT* yaitu fitur Add Monitor. Pada halaman ini terdiri dari beberapa isian form yaitu host address, *monitoring* category OID, *monitoring* interval, dan durasi dari waktu serta tanggal pengiriman proses *monitoring*. Setelah pengguna memilih tombol *submit* maka parameter perintah yang sudah di*set* oleh pengguna akan disimpan di dalam *database.* Perintah tersebut kemudian nanti akan diambil oleh *manager* untuk dikirimkan ke *agent*

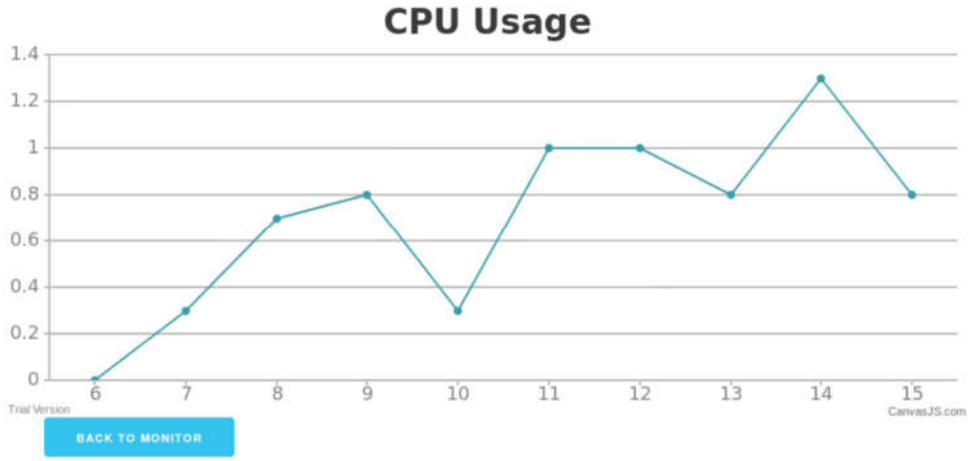
Source code 5.10 *Add Monitor*

```
@app.route('/add_monitor', methods=['POST'])
def add_monitor():
    _host = request.form['host']
    monitype = request.form['monitype']
    interval = request.form['interval']
    starttime = request.form['starttime'] + ':00.00'
    endtime = request.form['endtime'] + ':00.00'
    timestamp = datetime.datetime.now()
    if _host == 'All':
        all_host = models.Host.query.all()
        for i in range(len(all_host)):
            one_host = all_host[i]
            one_host.phase = "active"
            models.db.session.commit()
    else:
        active_host = models.Host.query.filter_by(host=_host).first()
        active_host.phase = "active"
        models.db.session.commit()
    query = models.Monitor(_host, monitype, interval, starttime, endtime,
timestamp)
    models.db.session.add(query)
    models.db.session.commit()
    # return all_host

    if monitype == '1' or monitype == '2' or monitype == '3' or monitype == '4' :
        return redirect('/res/'+ _host +'/'+ monitype)
    elif monitype == '5' or monitype == '6' :
        return redirect('/net/'+ _host +'/'+ monitype)
    elif monitype == '7':
        return redirect('/available')
    elif monitype == '8' :
        return redirect('/disk')
    elif monitype == '10' :
        return redirect('/res3/'+ _host +'/'+ monitype)
    elif monitype == '11' :
```

```
32          return redirect('/net3/'+ _host +'/'+ monitype)
33
```

Sesuai dengan yang ada pada *Source code* 5.10, pada fitur *add* monitor, hal yang pertama dilakukan adalah mendapatkan nilai yang diinputkan pada *form*. Pada baris 3 mengambil nilai dari host yang akan diamati yaitu MAC *address* yang sudah dipilih. Lalu pada baris 4 mengambil data berupa kategori OID yang diamati. Pada baris 5 mengambil nilai dari interval data *monitoring* yang dikirimkan dalam satuan detik. Pada baris 6 digunakan untuk mengambil tanggal dan jam mulai proses *monitoring*, sedangkan pada baris 7 untuk mengambil tanggal dan jam berhenti proses *monitoring*. Pada baris 8, perintah yang dimasukkan diberi tambahan atribut kapan perintah tersebut dibuat, hal ini digunakan untuk membuat perintah yang diambil *manager* adalalah data terakhir yang dimasukkan oleh pengguna.



Gambar 5.7 Halaman yang menampilkan CPU Usage

Pada gambar 5.7 menampilkan halaman yang menampilkan hasil *monitoring* yang tersimpan dalam tabel *resource*. Pengguna bisa melakukan filter berdasarkan host address dan kategori data *monitoring* yang ada. Data yang ada ditampilkan dalam bentuk grafik.

Source code 5.11 Get Resource

```
1   @app.route('/res/<host>/<category>')
2   def getHost(host, category):
3       host_list = models.Host.query.all()
4       host = models.Host.query.filter_by(host=host).options(load_only('host'))
5       if category == '1':
6           rows = models.Resource.query.filter_by(host=host).with_entities(
7               models.Resource.host, models.Resource.date,
8               models.Resource.time, models.Resource.cpu
9           ).all()
10
11          title = "CPU Usage"
12      elif category == '2':
13          rows = models.Resource.query.filter_by(host=host).with_entities(
14              models.Resource.host, models.Resource.date,
15              models.Resource.time,
16              models.Resource.memory_avail
17          ).all()
18          title = "Memory Available"
19      elif category == '3':
```

```
18          rows = models.Resource.query.filter_by(host=host).with_entities(
19              models.Resource.host, models.Resource.date,
20              models.Resource.time,
21              models.Resource.memory_used
22          ).all()
23          title = "Memory Used"
24      elif category == '4':
25          rows = models.Resource.query.filter_by(host=host).with_entities(
26              models.Resource.host, models.Resource.date,
27              models.Resource.time,
28              models.Resource.swap_free
29          ).all()
30          title = "Swap Free"
31      jarak = len(rows)-10
32      return render_template(
33          'res.html', rows=rows, host_list=host_list, host_id=host,
34          category=category, length=len(rows), title=title, jarak=jarak
35      )
```

Source code 5.11 memberikan gambaran bahwa pada baris 2-4 method gethost berfungsi untuk mendapatkan id host, id host dipakai untuk melakukan proses *filter* berdasarkan *host.* Lalu pada baris ke 6 dilakukan proses *filter* berdasarkan kategori *resource* yang ingin ditampilkan. Jika yang dipilih adalah 1 maka menampilkan hasil *monitoring cpu usage*, jika yang dipilih adalah 2 maka akan menampilkan hasil *monitoring memory available*, jika yang dipilih adalah 3 maka akan menampilkan hasil *monitoring memory used*, jika yang dipilih adalah 4 maka akan menampilkan hasil *monitoring swap free.*

Gambar 5.8 Halaman yang menampilkan Byte Traffic

Pada Gambar 5.8 menampilkan halaman yang menampilkan hasil *monitoring* yang tersimpan dalam tabel *network*. Pengguna bisa melakukan filter berdasarkan *host address* dan kategori data *monitoring* yang ada. Data yang ada ditampilkan dalam bentuk grafik.

Source code 5.12 *Get Network*

```
1   @app.route('/net/<host>/<category>')
2   def getAllNet(host, category):
3       host_list = models.Host.query.all()
4       host = models.Host.query.filter_by(host=host).options(load_only('host'))
5       if category == '5':
6           rows = models.Network.query.filter_by(host=host).with_entities(
7               models.Network.host, models.Network.byte_receive,
8               models.Network.byte_sent
9           ).all()
10          title = "Byte Traffic"
11          cat = ("Byte Sent", "Byte Receive")
```

```
12      elif category == '6':
13          rows = models.Network.query.filter_by(host=host).with_entities(
14              models.Network.host,
15              models.Network.packet_receive,
16              models.Network.packet_sent
17          ).all()
18          title = "Packet Traffic"
19          cat = ("Packet Sent", "Packet Receive")
20      jarak = len(rows)-10
21      return render_template(
22          'net.html', rows=rows, host_list=host_list, host_id=host,
23          category=category, length=len(rows), title=title, cat=cat, jarak=jarak
24      )
```

Source code 5.12 memberikan gambaran bahwa pada baris 2-4 *method getnet* berfungsi untuk mendapatkan id *host* untuk melakukan proses *filter* berdasarkan *host.* Lalu pada baris ke 6 dilakukan proses *filter* berdasarkan kategori *resource* yang ingin ditampilkan. Jika yang dipilih adalah 1 maka menampilkan hasil *monitoring byte traffic (byte sent* dan *byte receive)*, jika yang dipilih adalah 2 maka akan menampilkan hasil *monitoring packet traffic (packet sent* dan *packet receive).*

5.5 Implementasi pengiriman data

Implementasi pengiriman data yang dilakukan digunakan untuk jalur komunikasi antara *agent* dan *manager.* Implementasi dari proses pengiriman data oleh *agent* dapat dilihat pada Source code 5.13.

Source code 5.13 Proses Pengiriman data ke Manager

```
1   def getNetwork():
2       _type = 2
3       bytesent = psutil.net_io_counters().bytes_sent
4       byterecv = psutil.net_io_counters().bytes_recv
5       pktsent = psutil.net_io_counters().packets_sent
6       pktrcv = psutil.net_io_counters().packets_recv
7       net = {
8           'type': _type,
9           'mac': getmac(),
10          'bytesent': bytesent,
11          'byterecv': byterecv,
12          'pktsent': pktsent,
13          'pktrcv': pktrcv
14      }
15      result = json.dumps(net)
16      return result
17
18
19      def transporthandler(self, args):
20          if self.ACTION['SENDING'] == args[0]:
21              print 'sending...'
22          elif self.ACTION['WAITING'] == args[0]:
23              print 'waitting command'
24          self.transport.write(args[1])
25
```

Pada baris 8-13 method mengambil data yang dibutuhkan untuk dikirimkan. Pada baris ke 15, format data diubah ke dalam bentuk JSON. Pada baris 24, data kemudian dikirim melalui protokol UDP.

Source code 5.14 Proses Penerimaan Data dari Agent

```
def datagramReceived(self, data, (host, port)):
        handler = Handler(self.transport, host, port)
        result = json.loads(data)
        args = result.get('type') if result.get('type') else
result.get('hostname')
        d = threads.deferToThread(handler.handleMessage, args)
        db = sqlite3.connect("monitor.db")
        cursor = db.cursor()

        if result.get('type') == 1:
            cursor.execute('''INSERT INTO
resource(host,date,time,cpu,memory_avail,memory_used,swap_free)
            VALUES (?,?,?,?,?,?,?);''', (
                result.get('mac'), str(time.strftime("%d-%m-%Y")),
                str(time.strftime("%H:%M:%S")), result.get('cpu'),
result.get('memory'),
                result.get('usedmem'),
                result.get('swap')))
            print "resource has been inserted to database"
            print "result %r " % result
        elif result.get('type') == 2:
            cursor.execute(
                '''INSERT INTO
network(host,date,time,byte_sent,byte_receive,packet_sent,packet_receive)
                VALUES (?,?,?,?,?,?,?);''', (
                    result.get('mac'),
                    str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                    result.get('bytesent'),
                    result.get('byterecv'),
                    result.get('pktsent'),
                    result.get('pktrcv')
                )
            )
            print "network has been inserted to database"
            print "result %r " % (result)
        elif result.get('type') == 3:
            cursor.execute(
                '''INSERT INTO availability(host,date,time,status)
                VALUES (?,?,?,?);''',
                (result.get('mac'),
                 str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                 result.get('avai'))
            )
            print "this is availability"
            print "result %r " % (result)
            print type(result)
        elif result.get('type') == 4:
            cursor.execute(
                '''INSERT INTO
disk(host,date,time,disk_used,disk_free,read_bytes,write_bytes)
                VALUES (?,?,?,?,?,?,?);''', (
                    result.get('mac'), str(time.strftime("%d-%m-%Y")),
str(time.strftime("%H:%M:%S")),
                    result.get('disk_used'), result.get('disk_free'),
                    result.get('read_bytes'), result.get('write_bytes'))
            )
            print "disk has been inserted to database"
            print "result %r " % result
        elif result.get('type') == 5:
            print 'agent wait for command'
        else:
```

```
64              print "received from host %r , %s:%d" % (result.get('mac'), host,
65  port)
66              t = (result.get('mac'),)  # s = (result[1],)
67              query = cursor.execute('''SELECT * FROM Host WHERE host=?''', t)
68              count = len(query.fetchall())
69
70              if count == 0:
71                  cursor.execute(
72                      '''INSERT INTO host(hostname,host,status,phase) VALUES
73  (?,?,?,?);''',
74                      (result['hostname'], result['mac'], result['status'],
75  result['phase'])
76                  )
78                  print "insert"
79              else:
80                  cursor.execute(
81                      '''UPDATE Host SET status = ?, phase = ? WHERE hostname =
82  ?;''',
83                      (result['status'], result['phase'], result['hostname'])
84                  )
85                  print "update"
86              print "host has been recorded"
87          db.commit()
88          db.close()
89          d.addCallback(handler.postHandle)
90
```

Proses penerimaan data oleh *manager* dapat dilihat pada Source code 5.14. Data diterima oleh *Manager* melalui method *datagram Received*. Kemudian data diubah ke dalam bentuk *dictonary* kembali melalui baris 3. Setelah itu proses memasukkan data ke dalam *database* melalui proses *query*.