AITI: Jurnal Teknologi Informasi, Volume 16 No. 2 Agustus 2019, 151-164
ISSN 1693-8348 E-ISSN 2615-7128

JURNAL TEKNOLOGI INFORMASI

Analisis *Routing* Protokol *Optimized Link State Routing* (OLSR) Pada Raspberry Pi

Ulfa Septilia Permatasari [1], Indrastanti Ratna Widiasari [2]
[1,2]Fakultas Teknologi Informasi Universitas Kristen Satya Wacana
Jl. Dr. O. Notohamidjojo 1-10, Salatiga 50711, Indonesia
Email : [1]672011228@student.uksw.edu, [2] indrastanti@gmail.com

Abstract

Optimized Link State Routing (OLSR)is one of the two standard for mesh networks. OLSR is a link state routing protocol, which use hallo message and topology control (TC) to determine the link state information across the mobile ad-hoc network. OLSR is widely used in the wireless mesh network. Raspberry Pi is a mini computer which can be used as a node router to replace the function of a router in a wireless mesh network. In this research, Raspberry Pi is used as a router nodes to determine the performance of routing protocol OLSR. The parameters used to measure its performance is self-configure time, self-healing and bandwidth usage. The test results showed that the use of routing protocols OLSR on the Raspberry Pi in a wireless mesh network proved able to repair itself if there is a problem on the network since it has ability to self-configure and self-healing.

Keywords: *Optimized Link State Routing, Raspberry Pi, Wireless Mesh Network, Self-Configure, Self-Healing*

Abstrak

Optimized Link State Routing (OLSR) merupakan salah satu dari dua standart internet untuk jaringan *mesh.* OLSR *routing* protokol *link-state* yang menggunakan *message* hello dan *topologycontrol*(TC) untuk mengetahui informasi *link-state* di seluruh jaringan *mobile ad-hoc*. OLSR banyak digunakan pada *wireless mesh network.* Raspberry Pi yang merupakan mini komputer dapat digunakan sebagai node *router* untuk menggantikan fungsi dari router dalam *wireless mesh network*. Dalam penelitian ini, digunakan Raspberry Pi sebagai node *router* untuk mengetahui performansi dari *routing* protokol OLSR.Parameter yang digunakan untuk mengukur performansinya adalah waktu *self-configure*, *self-healing* dan penggunaan *bandwidth*. Penggunaan *routing* protokol OLSR pada Raspberry Pi dalam*wireless mesh network* terbukti dapat memperbaiki dirinya sendiri jika ada masalah pada perangkat jaringannya karena memiliki kemampuan *self-configure* dan *self-healing.*

Kata kunci: *Optimized Link State Routing,* Raspberry Pi,*Wireless Mesh Network*, *Self-Configure*, *Self-Healing*

1. Pendahuluan

Optimized Link State Routing (OLSR) adalah *routing* protokol *link-state* yang *proactive*, yang menggunakan *message* hello dan *topologycontrol*(TC) untuk mengetahui informasi *link-state* di seluruh jaringan *mobile ad-hoc*. OLSR banyak digunakan dan sudah teruji keandalannya karena sangat cepat dan menggunakan sedikit CPU *time* yang menyebabkan penggunaan baterai lebih sedikit dari *embedded* dan *portabledevice*. OLSR merupakan salah satu dari dua standart internet untuk jaringan *mesh*[1]. OLSR sering dijalankan pada *wireless mesh network*.

Wireless mesh network merupakan jaringan *mobilead-hoc* yang terkoneksi *peer-to-peer* ke jaringan. *Wireless mesh network*memiliki jangkauan yang luas karena setiap node tidak hanya bertindak sebagai *host* tetapijuga dapat berfungsi sebagai sebuah *router* untuk meneruskan paket-paket informasi yangakan dikirim menuju node lain yang mungkin tidak dapat menjangkau tempat yang inginditujunya karena keterbatasan jarak.*Wireless mesh network*memiliki kelebihan, dimana jika koneksi yang dilakukannya gagal, maka akan melakukan *routing* kembali dan dapat mengorganisasikan dirinya sendiri, sehingga koneksi tidak rentan putus.

Dalam penelitian ini digunakan *routing* protokol OLSR dan Raspberry Pi sebagai node. Perancangan jaringan dilakukan untuk pengujian terhadap performansi dari *routing* protokol OLSR pada Raspberry Pi dalam*Wireless mesh network.* Data yang diambil pada penelitian ini berupa waktu *self-configure* dan *self healing* yang dibutuhkan, selain itu juga penggunaan *bandwidth.*

Penelitian ini bertujuan untuk mengetahui performansi *routing* protokol OLSR pada Raspberry Pi sebagai node dalam w*ireless mesh network*. Dengan melihat parameter waktu *self-configure*, *self-healing* dan penggunaan *bandwidth* pada penelitian ini, maka dapat diambil kesimpulan sesuai dengan hasil yang didapatkan. Penelitian ini diharapkan dapat bermanfaat untuk bahan pertimbangan bahwa Raspberry Pi yang merupakan mini komputer dengan daya rendah juga dapat menjadi node *mesh* karena komponen utama *wireless mesh network* adalah suatu perangkat yang selain dapat berperan sebagai sumber trafik juga dapat berperan sebagai *router* yang mampu merutekan trafik dari sumber ke tujuan.

2. Tinjuan Pustaka

Pada penelitian lain membahas tentang implementasi protokol *routing* OLSR pada *Wireless Mesh Network* dengan *wireless* router TP-Link menggunakan OpenWRT Barrier Breaker dengan paket OLSR Daemon untuk menguji kinerja jaringan dengan parameter *self-configure, self-healing, jitter* dan *bandwidth*. Hasil yang didapatkan dari penelitian tersebut adalah protokol *routing* OLSR terbukti dapat memperbaiki dirinya sendiri jika ada masalah pada perangkat jaringan karena

memiliki kemampuan *self-healing* dan *self-configure*, sehingga jika ada perubahan topologi pada jaringan maupun ada jalur yang putus antara node masih bisa berkomunikasi. Pada pengujian *jitter* menunjukkan bahwa protokol OLSR memilih jalur tercepat walaupun harus melalui hop yang lebih banyak [3].

Berdasarkan penelitian terdahulu yang membahas tentang penggunaan protokol *routing* OLSR pada *wireless mesh network* menggunakan *hardware* yang berbeda yaitu *wireless* router LinkSys dan *wireless* router TP-Link, dengan *firmware opensource* OpenWRT yang hanya berbeda versi saja maka akan dilakukan penelitian tentang penggunaan *routing* protokol OLSR pada *wireless mesh network* menggunakan Raspberry Pi dengan sistem operasi Raspbian Jessie yang berbasis Debian ARM guna menguji performansinya. Parameter yang digunakan untuk menguji performansi adalah*self-healing, self-configure,*dan penggunaan *bandwidth,* karena routing protokol OLSR dalam *wireless mesh routing* memiliki karakteristik utama yaitu *self-configure* dan *self-healing* yang dapat mengkonfigurasi dan memperbaiki dirinya sendiri, dengan kata lain mampu menjaga konektivitas antar node apabila terjadi kerusakan pada salah satu node. Pengujian penggunaan *bandwidth* dengan kondisi jumlah *hop* berbeda, digunakanuntuk mendapatkan data tentang kinerja penggunaan *bandwidth* pada setiap node *mesh* dalam sistem *multihop* pada *wireless mesh network*..

Wireless mesh network adalah jaringan komunikasi terdiri dari node radio yang terorganisir dalam topologi *mesh*. *Wireless mesh network* terdiri dari dua node yaitu *mesh client* dan *mesh router*. Para *mesh client* biasanya *smartphone*, laptop, dan*gadget* yang memiliki fitur *wireless* lainnya sementara untuk *mesh router* meneruskan koneksi ke dan dari *gateway*atau dapat dikatakan sebagai jaringan *backbone*[4].Adapun karakteristik yang dimiliki oleh *Wireless Mesh Network* yang mempengaruhi kinerjanya adalah*multi-hop wireless network,* kemampuan *self-forming*, *self-healing*, *self-organizing* serta mendukung*ad-hoc networking*, tingkat mobilitas tergantung dari jenis node, dapat mengakses ke berbagai jenis teknologi jaringan lainnya, dependensi terhadap pemakaian daya tergantung dari jenis node [5].

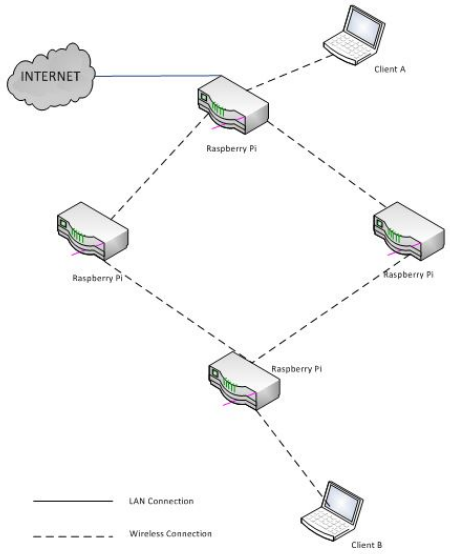
OLSR merupakan salah satu jenis dari *proactive routing protocol* yang biasa digunakan dalam jaringan *ad-hoc*. Protokol ini melakukan pertukaran pesansecara periodik dalam rangka menjaga informasi topologi jaringan yang ada pada setiap node.Setiap node mengirimkan paket kontrolnya masing-masing secara periodik sehingga dapat mentoleransi terjadinya *loss* dari beberapa paket pada saat-saat tertentu akibat dari tubrukan data ataupun akibat gangguan transmisi lainnya. OLSR menggunakan *multihop routing* dimana setiap node menggunakan informasi *routing* terbaru yang ada pada node tersebut dalam mengantarkan sebuah paket informasi.Secara umum OLSR memiliki langkah-langkah kerja, yaitu *link sensing, neighbour detection, MPR selection,* pengiriman TC (*Topology Control*) *messages,* dan *route calculation* [6].

3. Metode Penelitian

Penelitian yang dilakukan, diselesaikan melalui tahapan penelitian yang terbagi dalam empat tahapan, yaitu: (1) Menentukan topologi jaringan, (2) Menentukan spesifikasi perangkat, (3) Melakukan instalasi *software*, dan (4) Melakukan konfigurasi jaringan.

Gambar 1 Tahap - Tahap Penelitian

Tahap – tahap penelitian pada Gambar 1, dapat dijelaskan sebagai berikut. *Tahap pertama*: menentukan topologi jaringan. Hal ini dilakukan agar dapat menentukan dan membuat jaringan yang dibentuk oleh node *router* dan node *client*. Topologi jaringan yang dibuat disesuaikan dengan konsep *wireless mesh network*. Adapun topologi jaringan yang digunakan seperti pada Gambar 2.



Gambar 2 Perancangan Topologi Jaringan

Node *router* yang digunakan sebanyak empat buah agar dapat melakukan perpindahan jalur *routing* sesuai dengan cara kerja *wireless mesh network*. Salah

satu node *router* terhubung dengan internet menggunakan kabel, dan menjadi *gateway* bagi node *router* lain yang terhubung secara nirkabel agar dapat terkoneksi ke internet. Ada dua *client* yang digunakan, yaitu *client*A dan *client* B.

Tahap kedua: Penentuan spesifikasi perangkat dilakukan agar dapat mengetahui perangkat yang sesuai dan dapat berjalan dengan baik sesuai dengan kebutuhan *wireless mesh network.* Adapun perangkat yang digunakan dapat dispesifikasikan menjadi dua, yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*). Spesifikasi *hardware* berdasarkan topologi jaringan yang digunakan dalam penelitian, dibutuhkan empat buah node *router* (Raspberry Pi).

Wireless usb adapter juga dibutuhkan empat buah, yang digunakanuntuk dapat terhubung secara nirkabel pada jaringan. *Wireless* usb adapter yang digunakan harus mendukung mode *ad-hoc*, maka dari itu digunakan *wireless* usb adapter dengan *merk*TP-Link seri TL-WN722N. Spesifikasi *software* yang digunakan pada penelitian ini ada dua, yaitu *software* untuk node *router* dibutuhkan sistem operasi Raspbian Jessie 4.4 yang berbasis Debian ARM, aplikasi *routing* protokol OLSR Daemon (OLSRd) dan aplikasi untuk menguji performansi jaringan digunakan Iperf.

Tahap ketiga: Instalasi *software* dilakukan pada node *router* dan node *client,* agar *wireless mesh network* dapat berjalan dengan baik. Pada node *router* dilakukan instalasi Raspbian Jessie versi 4.4 yang merupakan sistem operasi berbasis Debian ARM yang khusus digunakan untuk perangkat keras Raspberry Pi, selain itu juga instalasi paket OLSRd yang merupakan *tools routing* protokol OLSR yang dirancang untuk mempermudah konfigurasi Raspberry Pi sebagai *wireless* node. Serta aplikasi Iperf yang digunakan untuk menguji performansi pada jaringan.

Tahap keempat: Konfigurasi jaringan merupakan hal yang terpenting dalam tahapan penelitian ini, dimana sangat menentukan berhasil atau tidaknya sistem. Beberapa hal yang perlu diperhatikan dalam mengkonfigurasi sebuah node*router* dan node *client* dalam suatu jaringan agar dapat bekerja dengan baik adalah pengalamatan dan konfigurasi *routing*. Pengalamatan harus dilakukan pada setiap*interface*node *router* dan node *client* dalam *wireless meshnetwork* dengan menggunakan Ipv4.Pengalamatan untuk *interface* node *router* pada LAN *interface* maupun *wirelessinterface*.

Kode Program 1 Pengalamatan *interface* pada Raspi 1

```
1.  auto lo
2.  iface lo inet loopback
3.  auto eth0
4.  iface eth0 inet dhcp
5.  auto wlan0
6.  iface wlan0 inet static
7.  address 10.10.10.1
8.  netmask 255.0.0.0
9.  wireless-mode ad-hoc
10. wireless-channel 1
```

```
11. wireless-essid MESH
12. iface default inet dhcp
```

Kode Program 2 Pengalamatan *interface* pada Raspi 2

```
1.  auto lo
2.  iface lo inet loopback
3.  auto eth0
4.  iface eth0 inet dhcp
5.  auto wlan0
6.  iface wlan0 inet static
7.  address 10.20.10.1
8.  netmask 255.0.0.0
9.  wireless-mode ad-hoc
10. wireless-channel 1
11. wireless-essid MESH
12. iface default inet dhcp
```

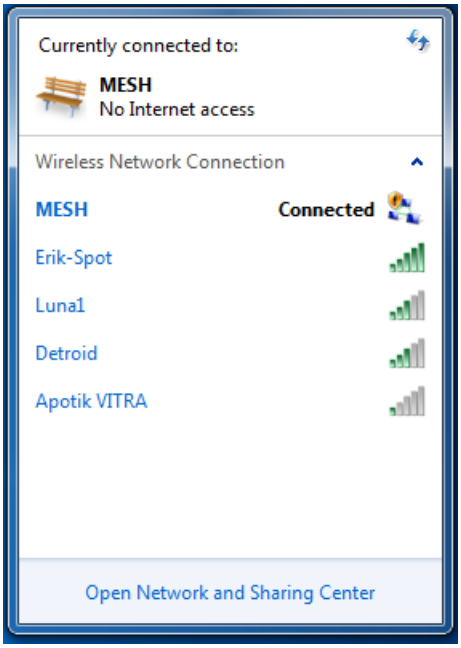
Kode Program 3 Pengalamatan *interface* pada Raspi 3

```
1.  auto lo
2.  iface lo inet loopback
3.  auto eth0
4.  iface eth0 inet dhcp
5.  auto wlan0
6.  iface wlan0 inet static
7.  address 10.30.10.1
8.  netmask 255.0.0.0
9.  wireless-mode ad-hoc
10. wireless-channel 1
11. wireless-essid MESH
12. iface default inet dhcp
```

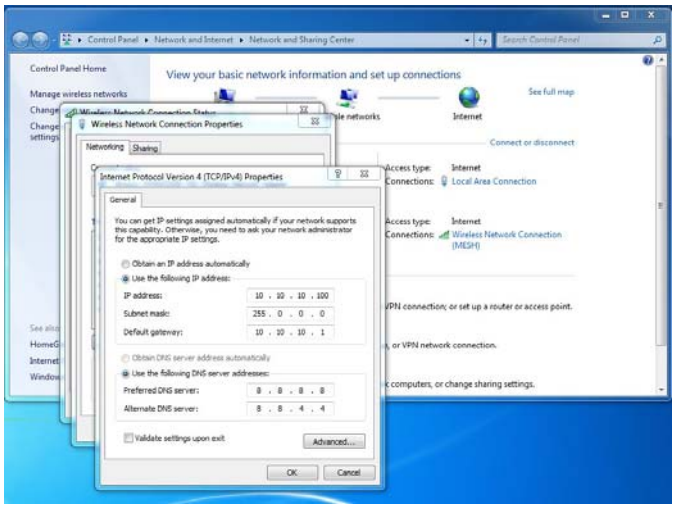
Kode Program 4 Pengalamatan *interface* pada Raspi 4

```
1.  auto lo
2.  iface lo inet loopback
3.  auto eth0
4.  iface eth0 inet dhcp
5.  auto wlan0
6.  iface wlan0 inet static
7.  address 10.30.10.1
8.  netmask 255.0.0.0
9.  wireless-mode ad-hoc
10. wireless-channel 1
11. wireless-essid MESH
12. iface default inet dhcp
```

Pengalamatan pada *interface* node *client* (laptop) ini secara *static* jika terhubung dengan*wireless meshnetwork* melalui koneksi *wireless*, yaitu dengan terkoneksi dengan *wireless meshnetwork* bernama ssid “MESH” dan kemudian mengubah alamat *pada interface wireless* laptop seperti pada Gambar 3 dan Gambar 4.



Gambar 3 *Ad-hocWireless Mesh Network*



Gambar 4 Pengalamatan *Interface* Pada Node *Client*

Konfigurasi *routing* dapat dilakukan dengan mengubah *file* olsrd.conf yang terdapat pada /etc/olsrd/olsrd.conf.

Kode Program 5 Konfigurasi *routing* pada Raspi

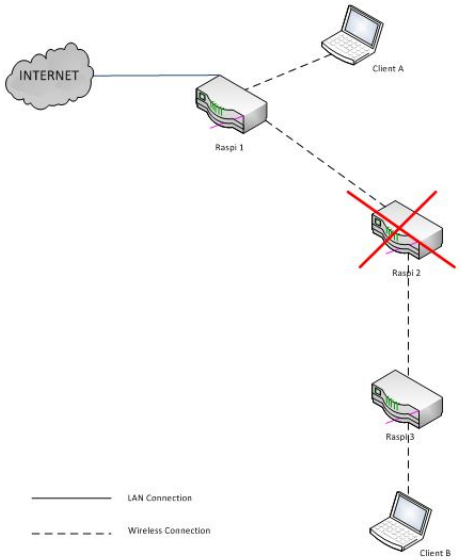
```
1.  DebugLevel        0
2.  OlsrPort          698
3.  RtTable           254
4.  RtTableDefault    254
5.  MainIp            10.10.10.1
6.  Interface "wlan0"
7.  {
8.  Mode "mesh"
9.  Ip4Broadcast      0.0.0.0
10. HelloInterval     5.0
```

```
11. HelloValidityTime 600.0
12. TcInterval                2.0
13. TcValidityTime    300.0
14. MidInterval       10.0
15. MidValidityTime   300.0
16. HnaInterval       10.0
17. HnaValidityTime   300.0
18. }
19. LinkQualityFishEye       1
20. LinkQualityAlgorithm "etx_ffeth"
21. IpVersion         4
22. ClearScreen yes
23. Hna4
24. {
25. 192.168.1.0 255.255.255.0
26. }
27. AllowNoInt yes
28. TosValue 16
29. Willingness
30. IpcConnect
31. {
32. MAxConnections 0
33. Host 127.0.0.1
34. }
35. UseHysteresis no
36. LInkQualityLevel  2
37. Pollrate          0.05
38. TcRedudancy       2
39. MprCoverage       5
40. LoadPlugin "olsrd_jsoninfo.so.0.0"
41. {
42. P1Param "accept" "127.0.0.1"
43. }
44. LoadPlugin "olsrd_httpinfo.so.0.1"
45. {
46. P1Param "Port "8080"
47. P1Param "Host" "192.168.1.0"
48. P1Param "Net" "192.168.1.0 255.255.255.0"
49. P1Param "Resolve" "true"
50. }
```

Beberapa konfigurasi yang penting adalah DebugLevel yang diberi nilai 0. DebugLevel merupakan fungsi kontrol yang mengatur jumlah *debug output* yang dapat ditampilkan, dengan nilai 0 maka proses yang dijalankan olsrd akan dijalankan dibelakang layar. IpVersion untuk menjelaskan jenis alamat IP yang digunakan pada jaringan, dalam penelitian ini jaringan menggunakan alamat IP versi 4. *Interface* untuk menunjukkan *interface* yang digunakan dalam jaringan *wireless*, yaitu "wlan0". Hna4 merupakan alamat LAN yang dapat berhubungan dengan node *router* dalam hal ini alamat jaringan 192.168.1.0 dengan *netmask* 255.255.255.0 sehingga setiap *client* yang terhubung melalui LAN akan mendapatkan alamat berdasarkan *subnet* yang sama. MainIp merupakan alamat *wireless interfaces* dari node *router* yang akan terhubung dengan node *router* lainnya, maupun node *client* yang terhubung melalui koneksi *wireless*, dalam hal ini alamat *wireless interfaces* node *router* 10.10.10.1. Nilai dari HelloInterval adalah 5.0 yang berarti pesan hello akan dikirimkan setiap 5 detik dan poin HelloValidityTime 600.0 berarti waktu valid informasi dalam pesan hello tersebut hingga 600 detik. Kemudian poin TcInterval 2.0 berarti pesan TC (*Topology Control*) akan dikirim setiap 2 detik dan poin TcValidityTime bernilai 300.0 berarti informasi pesan TC valid hingga 300 detik.

4. Hasil dan Pembahasan

Skenario untuk pengujian *self-configurerouting* protokol OLSR pada node *router* Raspberry Pi digunakan untuk mendapatkan data waktu yang diperlukan suatu node *router* untuk melakukan *self-configure* kepada dirinya sendiri dan bergabung dengan jaringan *wireless mesh* yang sudah ada. Pengujian dilakukan dengan mengubah variabel besarnya interval hello*message* dan TC *message* dari paket OLSR yang dapat diubah pada *file* etc/olsrd/olsrd.conf. Skenario pengujian *self-configure* dapat dilihat pada Gambar 5.



Gambar 5 Skenario Pengujian *Self-Configure*

Langkah-langkah yang dilakukan pada skenario pengujian *self-configure* adalah:

- *Client* A yang terhubung pada *wireless mesh network* melalui node raspi 1 melakukan hubungan dengan *client* B yang terhubung pada node raspi 2 dengan melakukan ping terus menerus.
- Node raspi 3 yang berada diluar jangkauan node raspi 1, karena node raspi 2 dimatikan maka ping akan menghasilkan *request time out*.
- Kemudian agar node raspi 1 dapat terhubung dengan node raspi 3, maka node raspi 2 dinyalakan.
- Kemampuan node raspi 2 untuk mengkonfigurasi inilah yang disebut dengan kemampuan *self-configure*.
- Data yang diambil adalah waktu mulai respon *reply*pertama pada *client*A yang melakukan ping terus menerus terhadap *client* B semenjak node raspi 2 dinyalakan.

Hasil dari pengujian *self-configure* dari *wireless mesh network*untuk nilai interval hello*Message* yang berbeda dapat dilihat pada Tabel 1.

Tabel 1 Tabel Pengujian *Self-Configure* Untuk Interval Hello*Message*

No	Hello Interval (detik)	TC Interval (detik)	Waktu (detik)
1	5	2	43.64
2	10	2	47.26
3	15	2	50.78
4	20	2	53.36
5	25	2	57.58
6	30	2	61.14

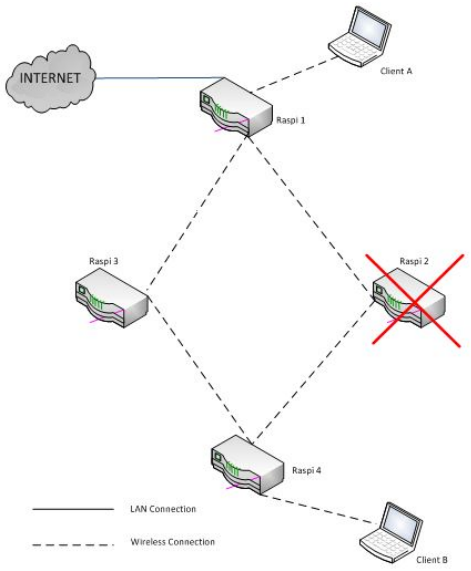
Hasil dari pengujian *self-configure* dari *wireless meshnetwork*untuk nilai interval TC (*Topology Control*) *Message* yang berbeda dapat dilihat pada Tabel 2.

Tabel 2 Tabel Pengujian *Self-Configure* Untuk Interval TC*Message*

No	Hello Interval (detik)	TC Interval (detik)	Waktu (detik)
1	5	2	42.76
2	5	5	45.84
3	5	10	48.46
4	5	15	50.94
5	5	20	53.96
6	5	25	56.34
7	5	30	59.22

Berdasarkan hasil pengujian yang dilakukan waktu rata-rata yang dibutuhkan untuk *self-configure*, didapatkan hasil bahwa besarnya nilai interval hello*message* sangat berpengaruh pada waktu yang dibutuhkan untuk melakukan *self-configure*. Waktu yang diperlukan untuk *self-configure* ikut bertambah seiring dengan bertambahnya nilai interval hello*message*. Sedangkan untuk besarnya nilai interval TC *message* tidak seberapa berpengaruh terhadap waktu yang dibutuhkan untuk melakukan *self-configure*.

Skenario untuk pengujian *self-healing routing* protokol OLSR pada node *router* Raspberry Pi digunakan untuk mendapatkan data waktu yang diperlukan jaringan untuk mencari rute baru maupun memperbaiki jalur apabila terjadi kerusakan jalur pada suatu node router. Pengujian dilakukan dengan mengubah variabel besarnya interval hello*message* dan TC *message* dari paket OLSR yang dapat diubah pada *file* etc/olsrd/olsrd.conf. Skenario pengujian *self-healing* dapat dilihat pada Gambar 6.



Gambar 6 Skenario Pengujian *Self-Healing*

Langkah-langkah yang dilakukan dalam skenario pengujian *self-healing*adalah :

- *Client* A pada node raspi 1 berhubungan dengan *client* B yang terhubung dengan node raspi 4 melalui jalur node raspi 2 dengan melakukan ping terus menerus.
- Kemudian node raspi 2 dimatikan, sehingga jalur node raspi 1 ke node raspi 4 melalui node raspi 2 tidak dapat dilalui sehingga ping akan menghasilkan *request time out.*
- Secara otomatis *routing* protokol OLSR akan mencari jalur baru melalui node raspi 3 yang masih menyala, inilah yang disebut *self-healing.*
- Data yang diambil adalah waktu mulai respon *reply* pertama *client* A pada node raspi 1 yang melakukan ping terus menerus untuk dapat berhubungan dengan *client* B pada node raspi 4 melalui jalur baru yaitu node raspi 3.

Hasil dari pengujian *self-healing* dari *wireless mesh network* untuk nilai interval hello m*essage* yang berbeda dapat dilihat pada Tabel 3.

Tabel 3 Tabel Pengujian *Self-Healing* Untuk Interval Hello*Message*

No	Hello Interval (detik)	TC Interval (detik)	Waktu (detik)
1	5	2	176.4
2	10	2	181.2
3	15	2	187.6
4	20	2	196.2
5	25	2	200.8
6	30	2	208.6

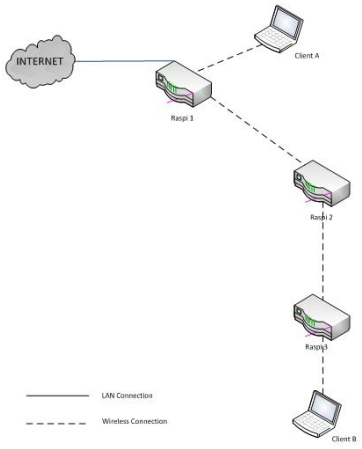
Hasil dari pengujian *self-healing* dari jaringan *wireless mesh* untuk nilai interval TC (*Topology Control*) *Message* yang berbeda dapat dilihat pada Tabel 4.

Tabel 4 Tabel Pengujian *Healing* Untuk Interval TC*Message*

No	Hello Interval (detik)	TC Interval (detik)	Waktu (detik)
1	5	2	175.8
2	5	5	179.4
3	5	10	181.8
4	5	15	182.4
5	5	20	184.6
6	5	25	185.2
7	5	30	187.8

Berdasarkan pengujian *self-healing* yang dilakukan pada *wireless mesh network*, sebuah node *router*akan secara otomatis mencari jalur sendiri jika terjadi kerusakan pada jalur yang dilalui sebelumnya. Ini terjadi karena routing protokol OLSR akan mendeteksi node tetangga (*Neighbour detection),* dimana disaat hello*message* dikirimkan tidak ada *update* informasi tentang suatu node. hello*message* yang tidak diterima lagi *update*-nya tentang suatu node dalam batas waktu tertentu akan menandakan bahwa hubungan terhadap node tersebut telah putus. Secara otomatis akan mencari jalur baru, dan *wireless mesh network* akan membentuk topologi baru. Pada *client*akan diketahui dengan munculnya *request time out* pada saat melakukan ping terus menerus.

Skenario untuk pengujian *bandwidth routing* protokol OLSR pada node *router* Raspberry Pi berfungsi untuk mendapatkan data pemakaian *bandwidth* pada masing-masing node *router* dalam sistem *multihop* pada jaringan *wirelessmesh*. Skenario pengujian dilakukan dengan variasi jumlah *hop* mulai dari 1 *hop* (R1-R2) dan 2*hop* (R1-R2-R3) dengan lima kali pengujian pada setiap skenario. Aplikasi Iperf diinstal pada ketiga node router, salah satu node dijadikan server dan node lain menjadi *client*. Skenario pengujian penggunaan *bandwidth* dapat dilihat pada Gambar 7.



Gambar 7 Skenario Pengujian *Bandwidth*

Hasil pengujian *bandwidth* dengan jalur 1 *hop* (R1-R2) dan 2 *hop* (R1-R2-R3) pada *wirelessmeshnetwork* dapat dilihat pada Tabel 5 dan Tabel 6.

Tabel 5 Tabel Pengujian *Bandwidth*Pada Jalur R1-R2

Uji ke-	Server - Client		Client - Server	
	Transfer (Mbytes)	**Bandwidth (Mbits/detik)**	**Transfer (Mbytes)**	**Bandwidth (Mbits/detik)**
1	17.2	14.9	17.9	15.1
2	15.9	13.5	16.2	13.6
3	17.0	15.0	18.0	15.1
4	14.9	13.2	15.9	13.3
5	17.2	15.1	18.2	15.2
Rata-rata	**16.44**	**14.34**	**17.24**	**14.46**

Tabel 6 Tabel Pengujian *Bandwidth* Pada Jalur R1-R2-R3

Uji ke-	Server - Client		Client - Server	
	Transfer (Mbytes)	**Bandwidth (Mbits/detik)**	**Transfer (Mbytes)**	**Bandwidth (Mbits/detik)**
1	7.3	6.1	7.5	6.3
2	7.9	6.3	8.2	6.8
3	7.3	6.1	7.5	6.3
4	8.2	6.8	8.6	7.2
5	7.7	6.6	8.4	7.1
Rata-rata	**7.68**	**6.38**	**8.04**	**6.74**

Berdasarkan hasil pengujian *bandwidth* antara dua jalur yaitu R1-R2 dengan R1-R2-R3 didapatkan hasil bahwa besarnya *bandwidth* dipengaruhi oleh banyaknya *hop* yang dilewati. Pada pengujian *bandwidth* dengan jalur R1-R2 (1 hop) didapatkan nilai rata-rata *bandwidth* sebesar 14.34 Mbit/s dengan besar transfer 16.44 Mbytes (Server-Client) dan 14.46 Mbit/s dengan besar transfer 17.24 Mbytes (Client-Server). Sedangkan untuk jalur R1-R2-R3 (2 hop) didapatkan nilai rata-rata *bandwidth* yang lebih kecil yaitu 6.38 Mbit/s dengan besar transfer 7.68 Mbytes (Server-Client) dan 6.74 Mbit/s dengan besar transfer 8.04 Mbytes (Client-Server).

5. Simpulan

Berdasarkan hasil analisis, perancangan dan pengujian kinerja *routing* protokol OLSR pada Raspberry Pi dalam*wireless mesh network*, yang telah dijelaskan pada bab sebelumnya, dapat diambil kesimpulan bahwa node *mesh* yang digunakan yaitu Raspberry Pi juga dapat menggantikan fungsi router yaitu mampu merutekan trafik dari sumber ke tujuan.Penggunaan *routing* protokol OLSR pada Raspberry Pi dalam*wireless mesh network* terbukti dapat memperbaiki dirinya sendiri jika ada masalah pada perangkat jaringannya karena memiliki kemampuan *self-configure* dan *self-healing*, besarnya nilai interval hello *message* lebih berpengaruh terhadap performansi *self-configure* pada *wireless mesh network*

dibandingkan besarnya nilai interval TC *message*, karena semakin besar nilai interval hello*message* maka waktu kerja *self-configure* semakin lama, nilai *bandwidth* pada jaringan *wireless mesh* sangat dipengaruhi oleh jumlah *hop* yang dilalui.

Saran yang dapat diberikan untuk penelitian dan pengembangan selanjutnya adalah memperhatikan keamanan pada *wireless mesh network* dan penggunaan jumlah node *router* dan *client* yang lebih banyak.

Daftar Pustaka

[1]. Purbo, Onno W., 2014, *Optimized Link State Protocol*(OLSR), http://cyberlearning.web.id/wiki/index.php/OLSR. Diakses pada 1 April 2016.

[2]. Mardani, Bagus, 2008,Analisis Unjuk Kerja *Wireless Mesh Network* Dengan *Routing Protocol* OLSR. Skripsi, Depok :JurusanTeknik Elektro Universitas Indonesia.

[3]. Setyawan, Adnan Puguh, 2014,Analisis Dan Perancangan *Wireless Mesh Networking* Menggunakan OLSR (*Optimized Link State Routing*) Berbasis OpenWRT Di Jogja Digital Valley. Skripsi, Yogyakarta :Jurusan Teknik Informatika Sekolah Tinggi Manajemen Informatika Dan Komputer AMIKOM Yogyakarta.

[4]. Akyildiz, Ian F., dkk., 2005, *Wireless Mesh Networks: a survey*, Georgia Institute of Technology.

[5]. Purbo, Onno W., 2013, Jaringan *MESH: S*olusi Jitu Membangun Jaringan *Wireless* Gotong Royong Tanpa *Access Point*, Andi Publisher.

[6]. Clausen, dkk., 2003, RFC 3626 :*Optimized Link State Protocol*(OLSR), https://www.ietf.org/rfc/rfc3626.txt. Diakses pada 1 April 2016.