Verifikasi Dua Varian Protokol Ad hoc On demand Distance Vector dengan UPPAAL

Ika Oktavia Suzanti*[1], M. Reza M.I. Pulungan[2]
[1] Prodi S2/S3 Ilmu Komputer, FMIPA UGM, Yogyakarta
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]i.oktavia.suzanti@gmail.com**, [2]pulungan@ugm.ac.id

Abstrak

Mobile Ad-hoc Network (MANET) adalah sekumpulan wireless mobile yang terhubung satu sama lain tanpa infrastruktur yang tetap sehingga perubahan topologi dapat terjadi setiap saat. Protokol routing MANET memiliki dua model yaitu protokol routing reaktif yang membentuk tabel routing hanya saat dibutuhkan dan protokol routing proaktif yang melakukan pemeliharaan tabel routing secara berkala. Properti umum yang harus dipenuhi oleh protokol jaringan ad-hoc adalah route discovery, packet delivery dan loop fredom.

AODV merupakan protokol reaktif MANET yang memiliki standar waktu berapa lama sebuah rute dapat digunakan (route validity), sehingga properti route discovery dan packet delivery harus dapat dipenuhi dalam waktu tersebut. Proses verifikasi protokol dilakukan dengan memodelkan spesifikasi protokol menggunakan teknik, tool, dan bahasa matematis.

Pada penelitian ini bahasa pemodelan yang digunakan adalah timed automata, yaitu bahasa pemodelan untuk memodelkan sistem yang memiliki ketergantungan terhadap waktu tertentu pada setiap prosesnya. Verifikasi protokol dilakukan secara otomatis dengan mengggunakan tool model checker UPPAAL.

Protokol yang diverifikasi adalah protokol AODV Break Avoidance milik Ali Khosrozadeh dkk dan protokol AODV Reliable Delivery dari Liu-Jian dan Fang-Min. Hasil verifikasi protokol membuktikan bahwa protokol AODV Break Avoidance mampu memenuhi properti route discovery dan protokol AODV Reliable Delivery mampu memenuhi properti packet delivery dalam waktu sesuai dengan spesifikasi.

Kata kunci *—Verifikasi Protokol, Timed Automata, AODV, UPPAAL*

Abstract

MANET is a group of wireless mobile that connected one to each other without fixed infrastructure so topology could change at anytime. MANET routing protocol has two models which are reactive routing protocol that built routing table only when needed and proactive routing protocol that maintain routing table periodically. General property which had to be satisfied by ad-hoc network protocol are route discovery, packet delivery and loop freedom.

AODV is a reactive protocol in MANET that has time standard to determine how long a route is valid to be used (route validity) so route discovery and packet delivery property should be satisfied in a specifically certain time. Protocol verification process done by modeling protocol specification using technique, tool and mathematic language.

In this research protocol modeled using timed automata which is a modeling language that could be used to model a time dependent system in each process. Verification using timed automata can automatically done by UPPAAL tool model checker.

Protocol which will be verified are AODV Break Avoidance by Ali Khosrozadeh et al. and AODV Reliable Delivery by Liu Jian and Fang-Min. Result of this protocol verification prove that AODV BA could satisfied route discovery property and AODV Reliable Delivery could satisfied packet delivery property within their specification time.

Keywords*—Protocol Verification, Timed Automata, AODV, UPPAAL*

Received February 3rd,2014; Revised January 8th, 2015; Accepted January 20th, 2015

1. PENDAHULUAN

Mobile Ad Hoc Network (MANET) adalah sekumpulan wireless mobile yang terhubung satu sama lain secara dinamis tanpa membutuhkan infrastruktur jaringan yang tetap [1]. Setiap node yang terhubung pada MANET tidak hanya dapat berperan sebagai client namun juga dapat berperan sebagai router yang bebas bergerak dari satu tempat ke tempat yang lain dalam jangkauan. Permasalahan yang utama pada MANET adalah adanya mobilitas pada setiap node karena terjadinya perubahan topologi pada MANET setiap saat sehingga dibutuhkan sebuah mekanisme routing yang dapat menangani dan menjamin sampainya paket data ke node tujuan yang benar dan tanpa waktu *delay* yang signifikan[1]. [2] memberikan spesifikasi umum yang seharusnya dipenuhi oleh sebuah protokol *routing* pada jaringan ad hoc yaitu apabila terdapat path antara dua buah node dalam jaringan maka protokol harus dapat menemukan rute diantara keduanya (*route discovery*) dan ketika rute tersebut ditemukan maka kedua *node* tersebut akan dapat saling berkomunikasi dalam jaringan (*packet delivery*).

AODV [1] merupakan salah satu protokol routing reaktif yang banyak digunakan dan dikembangkan sehingga berbagai macam studi terus dilakukan untuk memperbaiki serta meningkatkan kinerja protokol AODV. Protokol AODV merupakan protokol yang memiliki standar untuk menentukan berapa lama waktu sebuah rute dapat digunakan (*route validity*), karena itu proses *route discovery* harus selesai pada waktu tertentu sesuai dengan standart tersebut. Untuk mengurangi adanya interval waktu yang digunakan untuk memilih jalur alternatif karena menunggu jalur utama yang rusak diperbaiki dan tidak adanya mekanisme yang digunakan untuk memilih *node* alternatif jika terdapat lebih dari satu *node* alternatif [3] menawarkan solusi yang disebut AODV Reliable Delivery (AODV-RD). AODV-RD memiliki mekanisme link failure untuk memprediksi terjadinya kerusakan pada jalur utama, sehingga dapat mengurangi interval waktu antara rusaknya jalur utama dan pemilihan jalur alternatif pada proses pengiriman data. AODV Break Avoidance (AODV-BA) [4] adalah salah satu penelitian yang digunakan untuk memperbaiki AODV tradisional yang dianggap kurang baik karena harus membuat rute ulang ketika terjadi kerusakan pada jalur utama, oleh karena itu AODV-BA menawarkan algoritma perawatan rute dari kerusakan sehingga akan selalu ada rute diantara setiap node.

Timed Automata adalah salah satu varian bahasa pemodelan yang dapat digunakan untuk memodelkan sistem yang memiliki ketergantungan terhadap waktu tertentu pada setiap prosesnya. Verifikasi dengan timed automata dapat dilakukan secara otomatis dengan mengggunakan *tool model checker* UPPAAL. Penelitian yang dilakukan oleh [5] dan [6] menyatakan bahwa AODV tradisional yaitu AODV draf 13 milik [1] tidak memenuhi properti *route discovery* dan *packet delivery*. Sehingga pada penelitian kali ini akan dilakukan verifikasi pada protokol AODV RD dan AODV BA yang diharapkan mampu memenuhi protokol *route discovery* dan *packet delivery*

2. METODE PENELITIAN

2.1 Timed Automata

Timed Automata (TA) adalah sebuah mesin otomata dengan jumlah *state* berhingga (*finite state automata*) yang memiliki variabel clock. *Timed automata* digunakan untuk memodelkan sebuah sistem yang memiliki ketergantungan terhadap fungsi waktu dan memformulasikan keadaan *real time* dari sebuah sistem tersebut dalam bentuk program graf. Definisi formal dari *timed automata* dapat didefinisikan sebagai berikut :

Definisi 1. Sebuah *timed automaton* adalah sebuah himpunan yang terdiri dari :

$$TA = (Loc, Act, C, \rightarrow, Loc0, Inv, AP, L)$$

Dimana *Loc* adalah himpunan dari *locations*, *Loc*0 ⊆ *Loc* adalah himpunan inisial *locations*, *Act* adalah himpunan *actions*, *C* adalah himpunan *clocks*, → ⊆ *Loc* × *CC*(*C*) × *Act* × 2*C* × *Loc* adalah fungsi transisi, *Inv* : *Loc* → *CC*(*C*) adalah fungsi *invariant-assignment*, *AP* adalah himpunan dari *atomic preposition*, *L* : *Loc* → 2*AP* adalah *labeling function* dari *location*, *ACC (TA)* adalah himpunan *clock constraint* yang muncul pada *guard* atau *invariant* di sebuah *location* pada *Timed Automata* [7].

Perpindahan *location* dari ℓ ke ℓ' dilakukan apabila *guard* g terpenuhi $\ell \xrightarrow{g:\alpha,C} \ell'$. Pada saat transisi tersebut dilaksanakan maka clock C di-*reset* dan dilakukan *action* α yang merupakan proses sinkronisasi diantara dua atau lebih *timed automata* yaitu proses mengirim data (a!) dan menerima data (a?). Pemodelan sebuah sistem dengan automata umumnya dilakukan dengan menjalankan dua atau lebih *timed automata* secara *parallel*, diantara automata tersebut terdapat beberapa siknronisasi atau *handshaking* yang dilakukan untuk memodelkan pertukaran pesan maupun data tertentu. Operator yang digunakan untuk memodelkan *handshaking* pada *timed automata* adalah $\|_H$ dimana H adalah himpunan dari semua *action handshaking* yang dilakukan.

Definisi 2. Definisi *formal* dari *handshaking timed automata* adalah apabila terdapat *timed automata* TAi = (Loc_i, Act_i, C_i, →, Loc_0,i, Inv_i, AP_i, L_i), i = 1, 2 dengan H ⊆ $Act_1 \cap Act_2$, $C_1 \cap C_2 = 0$ dan $AP_1 \cap AP_2 = 0$ maka $TA_1 \|_H TA_2$ jika

($Loc_1 \times Loc_2$, $Act_1 \cup Act_2$,$C_1 \cup C_2$,→, $Loc_{0,1} \times Loc_{0,2}$, Inv , $AP_1 \cup AP_2$, L)

dimana dan Relasi transisi → didefinisikan dengan aturan sebagai berikut.

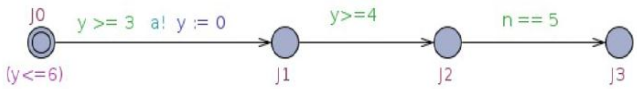
Untuk α ∈ H :

$$\frac{\ell_1 \xrightarrow{g_1:\alpha_1,C_1} \ell'_1 \wedge \ell_2 \xrightarrow{g_2:\alpha_2,C_2} \ell'_2}{<\ell_1,\ell_2> \xrightarrow{g_1 \wedge g_2:\alpha,C_1 \cup C_2} <\ell'_1,\ell'_2>}$$

untuk α ∉ H :

$$\frac{\ell_1 \xrightarrow{g:\alpha,C} \ell'_1}{<\ell_1,\ell_2> \xrightarrow{g:\alpha,C} <\ell'_1,\ell_2>} \text{ dan } \frac{\ell_2 \xrightarrow{g:\alpha,C} \ell'_2}{<\ell_1,\ell_2> \xrightarrow{g:\alpha,C} <\ell_1,\ell'_2>}$$

Untuk action α Є H, transisi dari pada *timed automata* dapat dilakukan hanya pada saat *guard* dari kedua transisi yaitu *guard* untuk *action* a! dan a? sama-sama dapat terpenuhi. Gambar 1 adalah contoh *timed automata* sederhana.



Gambar 1 Contoh *timed automata*

2. 2. Timed Computational Tree Logic (TCTL)

Timed computational tree logic (TCTL) adalah turunan dari computational tree logic (CTL).TCTL digunakan untuk memodelkan properti dari *timed automata* ke dalam bentuk logika *temporal* sehingga dapat dilakukan verifikasi yaitu dengan mencocokkan apakah TCTL tersebut mampu dipenuhi oleh *transition system* dari *timed automata.*

Sintaks TCTL adalah sebuah *query language* yang gabungan dari *state* formula, operator *boolean* dan *path* formula. *State* formula mendeskripsikan sebuah *location* dan *path* formula mendeskripsikan sebuah *edge*. *Boolean* operator yang dapat digunakan pada TCTL adalah berupa *true, false, and* (^), →, ↔ dengan *quantifier* ∃ dan ∀ serta *temporal* operator *eventually* (◊) dan *always* (□).Sintaks TCTL pada UPPAAL dengan *state formula* Φ dan ψ dapat berupa :

∀ □ *Φ* yang artinya untuk semua *path* maka Φ selalu dapat dipenuhi

$\exists \Box \Phi$ yang artinya terdapat *path* dimana Φ selalu dapat dipenuhi
$\forall \Diamond \Phi$ yang artinya untuk semua *path* maka Φ *eventually* dapat dipenuhi
$\exists \Diamond \Phi$ yang artinya terdapat *path* dimana Φ *eventually* dapat dipenuhi
$\Phi \rightarrow \psi$ yang artinya apabila Φ terpenuhi maka ψ juga akan terpenuhi

2. 3 Ad hoc On demand Distance Vector Protocol (AODV)

AODV merupakan salah satu protokol routing reaktif yang mampu beradaptasi dengan cepat terhadap perubahan topologi yang dinamis dalam jaringan ad hoc [1]. Proses pembentukan *route* pada protokol *routing* AODV yaitu dengan menggunakan dua pesan yaitu dengan menggunakan RREQ dan RREP [1]. Pada saat sebuah *node* (*source node*) akan mengirimkan paket atau data kepada *node* (*destination node*) lain dalam jaringan tapi dalam tabel *routing*-nya belum ada informasi menuju *node* tujuan maka *node* tersebut akan menginisialisasi proses *route discovery* untuk menemukan rute ke *node* tujuan dengan langkah berikut :

1. *Source node* akan mem-*broadcast* paket RREQ ke *node* tetangganya. RREQ paket berisi *source address*, *destination address*, *hop counter*, *source* and *destination sequence number*, dan *broadcast* ID. Nilai *broadcast* ID bertambah satu setiap suatu *source node* mengirimkan RREQ yang baru dan digunakan sebagai identifikasi sebuah paket RREQ.
2. Jika *node* yang menerima RREQ memiliki informasi rute menuju *node* tujuan, maka *node* tersebut akan mengirim paket RREP kembali menuju *source node*. Tetapi jika tidak memiliki informasi rute maka *node* tersebut akan mem-*broadcast* ulang RREQ ke *node* tetangganya setelah menambahkan nilai *hop counter*.
3. Node yang menerima RREQ dengan nilai *source address* dan *broadcast* ID yang sama dengan RREQ yang diterima sebelumnya akan membuang RREQ tersebut. *Source sequence number* digunakan oleh suatu *node* untuk memelihara informasi yang valid mengenai *reverse path* (jalur balik) menuju ke *source node*. Pada saat RREQ dikirim menuju *node* tujuan yang diinginkan, dia akan menciptakan *reverse path* menuju *source node*, setiap *node* akan membaca RREQ dan mengidentifikasi alamat dari *node* tetangga yang mengirim RREQ tersebut.
4. Ketika *node* tujuan atau *node* yang memiliki informasi rute menuju *destination* menerima RREQ maka *node* tersebut akan membandingkan nilai *destination sequence number* yang dia miliki dengan nilai *destination sequence number* yang ada di RREQ.
5. RREP akan dikirim menuju *source node* apabila nilai *destination sequence number* yang ada di *node* lebih besar atau sama dengan nilai yang ada di RREQ. Berbeda dengan RREQ, RREP dikirim secara *unicast*.

Pada saat pengiriman data apabila terjadi perubahan topologi yang mengakibatkan suatu *node* tidak dapat dituju dengan menggunakan informasi rute yang ada pada *routing table*, maka suatu node akan mengirim pesan RRER ke *node* tetangganya dan *node* tetangganya akan mengirim kembali RRER yang sama ke *node* tetangga yang lain, dan begitu seterusnya hingga menuju *source node*. Setiap *node* yang memperoleh RRER ini akan menghapus informasi yang mengalami *error* di dalam *routing table*-nya. Kemudian *source node* akan melakukan proses *route discovery* kembali apabila rute tersebut masih diperlukan .

Untuk mengurangi adanya interval waktu yang digunakan untuk memilih jalur alternatif karena menunggu jalur utama yang rusak diperbaiki dan tidak adanya mekanisme yang digunakan untuk memilih *node* alternatif jika terdapat lebih dari satu *node* alternatif [3] menawarkan solusi yang disebut AODV Reliable Delivery (AODV-RD). AODV-RD memiliki mekanisme *link failure* untuk memprediksi terjadinya kerusakan pada jalur utama, sehingga dapat mengurangi interval waktu antara rusaknya jalur utama dan pemilihan jalur alternatif pada proses pengiriman data.

AODV Break Avoidance (AODV-BA) [4] adalah salah satu penelitian yang digunakan untuk memperbaiki AODV tradisional yang dianggap kurang baik karena harus membuat rute ulang ketika terjadi kerusakan pada jalur utama. Algoritma yang diusulkan oleh AODV-BA adalah algoritma avoiding link break yaitu setiap *node intermediate* memberikan informasi pada *node* yang berada satu *hop* setelahnya *(node* tetangganya) bahwa terdapat paket yang akan dikirim melalui *node* tersebut. Apabila dalam waktu tertentu *node* tetangga tidak memberikan respon atas RREQ yang dikirim maka *node* tersebut mendeteksi adanya kerusakan *link* (*link break*) pada jalur tersebut. Deteksi ini diberitahukan pada *node* sebelumnya dengan cara men-*drop* sementara paket yang akan melewati *node* yang terdeteksi *link break* serta melakukan pembuatan jalur baru sebelum rute utama benar-benar rusak [4]. Pendeteksian ini dilakukan dengan mengukur tingkat kekuatan sinyal dari paket yang diterimanya.

Mekanisme perbaikan pada AODV BA ini pada dasarnya hampir mirip dengan mekanisme pada AODV RD namun yang menjadi pembedanya adalah apabila pada AODV BA jarak antara *node* yang mendeteksi *link break* ke *node* tujuan lebih dekat daripada jarak antara *node intermediate* menuju *node* pengirim maka baru kemudian dilakukan mekanisme *avoiding link break* akan tetapi sebaliknya jika jarak antara *node intermediate* yang mendeteksi *link break* lebih dekat ke *node* pengirim maka *node intermediate* tersebut akan mengirim RERR pada *node* pengirim untuk kemudian dilakukan pengulangan proses *route discovery.* Adapun pada AODV RD, pertimbangan jarak tersebut diabaikan sehingga dimanapun letak *node intermediate* yang mendeteksi *link failure* maka *node* tersebut akan mengirimkan RERR pada node pengirim sehingga node pengirim dapat segera melakukan pengulangan prose *route discovery.*

2.4 Formalisasi properti *route discovery* dan *packet delivery*

2.4.1 *Route discovery*

Properti *Route Discovery* adalah sebuah properti pada protokol AODV yang memiliki arti bahwa apabila terdapat sebuah *path* diantara dua buah *node* pada jaringan maka protokol akan dapat menemukan rute diantara kedua *node* tersebut. Protokol AODV merupakan protokol pada MANET yang memiliki sifat *On Demand* yaitu rute antara *node* pengirim dan *node* penerima hanya akan dibangun pada saat dibutuhkan agar jaringan tidak mengalami *overhead* karena banyaknya pesan yang harus dikirim. *Node* pada sebuah jaringan *ad hoc* dapat berjumlah banyak dimana masing-masing *node* tersebut dapat menjadi *node* pengirim, *node* penerima dan *node* perantara sehingga menyebabkan perubahan topologi jaringan dalam waktu tertentu. Protokol AODV merupakan protokol yang memiliki standar untuk menentukan berapa lama waktu sebuah rute dapat digunakan (*route validity*), karena itu proses *route discovery* harus selesai pada waktu tertentu sesuai dengan standart tersebut. Dengan kata lain apabila dalam waktu yang ditentukan setelah *node* pengirim mem-*broadcast* RREQ pada *node intermediate, node* pengirim tidak menerima RREP maka RREQ tersebut akan dibatalkan dan mengirim ulang RREQ yang baru. Apabila *node* pengirim telah menerima RREP atas RREQ yang dikirim sebelumnya maka *timed automata* dari *node* pengirim akan berada pada *location* recv_rrep, sintaks TCTL dari properti *route discovery* ditunjukkan pada persamaan (1) atau persamaan(2).

$$\exists \Diamond \text{ Pengirim.got\_route} \wedge \text{Pengirim.rreq\_timer} \leq \text{Pengirim.NETTT} \quad (1)$$

$$\forall \Diamond \text{ Pengirim.got\_route} \wedge \text{Pengirim.rreq\_timer} \leq \text{Pengirim.NETTT} \quad (2)$$

Quantifier ∃ pada persamaan (1) memiliki arti terdapat *path* sedangkan *quantifier* ∀ pada persamaan (2) memiliki arti untuk semua *path* dimana setelah waktu tertentu (*eventually*) *node* pengirim akan menemukan rute menuju *node* penerima yaitu berada pada *location* got_route dan rreq_timer dari *node* pengirim kurang atau sama dengan net traversal time.

2.4.2 *Packet delivery*

Properti *packet delivery* adalah properti yang menunjukkan bahwa apabila telah terdapat jalur diantara dua buah *node* pada jaringan maka kedua *node* tersebut dapat saling berkomunikasi maupun mengirimkan pesan. Properti *Packet Delivery* merupakan properti yang akan dapat terpenuhi hanya jika properti *route discovery* terlebih dahulu dipenuhi. Secara *intuitive,* hal ini menunjukkan bahwa properti *packet delivery* merupakan *liveness property* dari

reachability property route discovery. Adapun sintak TCTL untuk properti packet delivery ditunjukkan pada persamaan (3) atau persamaan (4).

Persamaan (3) memiliki arti terdapat *path* (∃) dan persamaan (4) memiliki arti semua path (∀) *eventually* dimana setelah waktu tertentu (*eventually*) *node* penerima telah menerima yaitu berada pada *location* recv_data dan rreq_timer dari *node* pengirim kurang atau samadengan active route time (ART). *Timed automata node* pengirim berada pada *location* recv_data sebagai tanda bahwa data yang dikirim oleh *node* pengirim telah sampai pada *node* penerima.

$$\exists \Diamond \text{ Penerima.recv\_data} \wedge \text{Pengirim.rreq\_timer} \leq \text{Pengirim.ART} \quad (3)$$

$$\forall \Diamond \text{ Penerima.recv\_data} \wedge \text{Pengirim.rreq\_timer} \leq \text{Pengirim.ART} \quad (4)$$

2.5 Pemodelan Protokol dengan timed automata

Pada pemodelan protokol AODV dengan UPPAAL diberikan beberapa asumsi untuk membatasi dan menentukan model yang dibuat yaitu

1. Model yang dibuat terdiri dari satu *node* pengirim, satu *node* penerima, n *node intermediate* dan pada saat simulasi dilakukan tidak ada *node* yang hilang. Pemilihan model ini mengacu pada penelitian yang dilakukan [6] dengan modifikasi pada proses *route discovery* dan penambahan proses *route maintenance* sesuai dengan spesifikasi protokol AODV Reliable Delivery dan AODV Break Avoidance.
2. Pada model yang dibuat, *node* pengirim memiliki alamat IP 0, *node* penerima memiliki alamat IP N-1 dimana N adalah jumlah semua *node* pada jaringan, dan *node intermediate* memiliki alamat IP 1 sampai N-2 dengan asumsi semakin mendekati N-1 maka *node intermediate* tersebut semakin dekat dengan *node* penerima.
3. Pengiriman ulang RREQ oleh *node intermediate* kepada *node intermediate* lainnya hanya dapat diterima oleh *node intermediate* yang memiliki IP lebih besar dari *node intermediate* yang mengirim ulang RREQ. Hal ini dilakukan agar tidak terjadi *loop* pada saat proses simulasi dijalankan.
4. *Node* yang dapat menginisialisasi RREP sebagai respon atas RREQ yang dikirim hanya *node* penerima.

2.5.1 *Handshaking timed automata*

Timed automata yang dibuat merupakan sinkronisasi dari *node* pengirim, beberapa *node intermediate* dan *node* penerima. Secara formal dapat didefinisikan sebagai berikut

$$\text{Pengirim} \parallel_H \text{Intermediate} \parallel_H \text{Penerima}$$

dimana *node intermediate* terdiri dari *node intermediate* ke 1 hingga *node intermediate* ke N yang secara *formal* didefinisikan :

$$\text{Intermediate}_1 \parallel_H \cdots \parallel_H \text{Intermediate}_n$$

Handshaking pada *timed automata* tersebut terjadi pada saat proses *route discovery* dan pengiriman data. Pada saat proses *route discovery* terjadi beberapa *handshaking* yaitu pada pengiriman RREQ, RREP dan RERR apabila jalur dari *node* pengirim menuju *node* penerima tidak dapat dibentuk.

Berdasarkan definisi 2 maka secara *formal* sintak proses pengiriman RREQ dari *node* pengirim pada *node intermediate* ditunjukkan pada persamaan (5). Pada persamaan (5) secara detail, transisi keluar *timed automata node* pengirim dari *location* idle ke *location* sent_rreq akan dilakukan secara bersamaan dengan transisi keluar *timed automata node intermediate* dari *location* idle ke *location* recv_rreq, dengan I dan J adalah *assignment* yang harus dilakukan pada saat proses tersebut dilakukan. Apabila jumlah *node intermediate* adalah n maka pada pengiriman RREQ, *handshaking* pada *node intermediate* terjadi antara *node intermediate* ke i dan *node intermediate* i+1. Adapun secara formal, sintak pengiriman RREQ dari *node intermediate* ke *node intermediate* lainnya ditunjukkan pada persamaan (6). Pada persamaan (6)

ditunjukkan bahwa pengiriman ulang RREQ ini dapat diterima oleh *node intermediate* lainnya dengan syarat *node intermediate* penerima memiliki IP lebih besar atau samadengan dari nilai variabel ips yaitu IP dari *node intermediate* pengirim ulang RREQ ditambah. Sedangkan sintak pengiriman RREQ dari *node intermediate* ke *node* penerima ditunjukkan pada persamaan (7). Pada persamaan (7), transisi keluar *timed automata node intermediate* dari *location* process_rreq kembali ke *location* idle akan dilakukan secara bersamaan dengan transisi keluar *timed automata node* penerima dari *location* idle ke *location* recv_rreq, dengan J adalah *assignment* yang harus dilakukan pada saat proses tersebut dilakukan. Sedangkan sintak pengiriman RREP dari node penerima ke *node intermediate* ditunjukkan pada persamaan (8). Adapun sintak pengiriman RREP dari *node intermediate* pada *node intermediate* lainnya ditunjukkan pada persamaan (9) dan sintak pengiriman RREP dari *node intermediate* pada *node* penerima ditunjukkan pada persamaan (10).

$$\frac{Pengirim.Idle \xrightarrow{status\_route==0\ RREQ!I} Pengirim.sent\_rreq \wedge Inter.Idle \xrightarrow{ips\leq me\ \ RERQ?J} Inter.recv\_rreq}{< Pengirim.Idle, Inter.Idle > \xrightarrow{status_{route}==0 \wedge ips\leq me\ \alpha I,J} < Pengirim.sent\_rreq, Inter.recv\_rreq >} \quad (5)$$

$$\frac{Interi.process\_rreq \xrightarrow{RREQ!} Interi.Idle \wedge Inter(i+1).Idle \xrightarrow{ips\leq me\ \ RREQ?J} Interi+1.recv\_rreq}{< Interi.process\_rreq,\ Interi+1.Idle > \xrightarrow{ips\leq me\ \ \alpha J} < Interi.Idle, Interi+1.recv\_rreq >} \quad (6)$$

$$\frac{InterN.process\_rreq \xrightarrow{RREQ!} InterN.Idle \wedge Penerima.Idle \xrightarrow{ips==me\ \ RREQ?J} Penerima.recv\_rreq}{< InterN.process\_rreq,\ Penerima.Idle > \xrightarrow{ips==me\ \ \alpha J} < InterN.Idle, Penerima.recv\_rreq >} \quad (7)$$

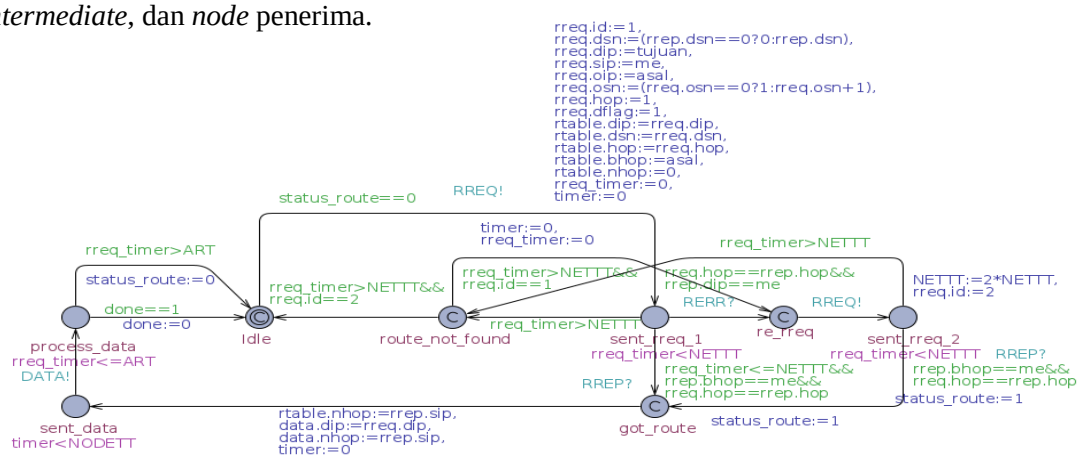
$$\frac{Penerima.sent\_rrep \xrightarrow{RREP!} Penerima.Idle \wedge Interi.Idle \xrightarrow{rrep.bhop==me\ \ RREP?J} Interi.recv\_rrep}{< Penerima.sent\_rrep,\ Interi.Idle > \xrightarrow{rrep.bhop=me\ \ \alpha J} < Penerima.Idle, Interi.recv\_rrep >} \quad (8)$$

$$\frac{Inter(i+1).process\_rrep \xrightarrow{rrep.signal\ \&rrep.hop,\ RREP!} Inter(i+1).Idle \wedge Interi.idle \xrightarrow{rrep.bhop==me\ \ RREP?J} Interi.recv\_rrep}{< Inter(i+1).process\_rreq,\ Interi.Idle > \xrightarrow{rrep.signal\ \&rrep.hop,\ \ rrep.bhop==me\ \ \alpha,\ J} < Inter(i+1).Idle, Interi.recv\_rrep >} \quad (9)$$

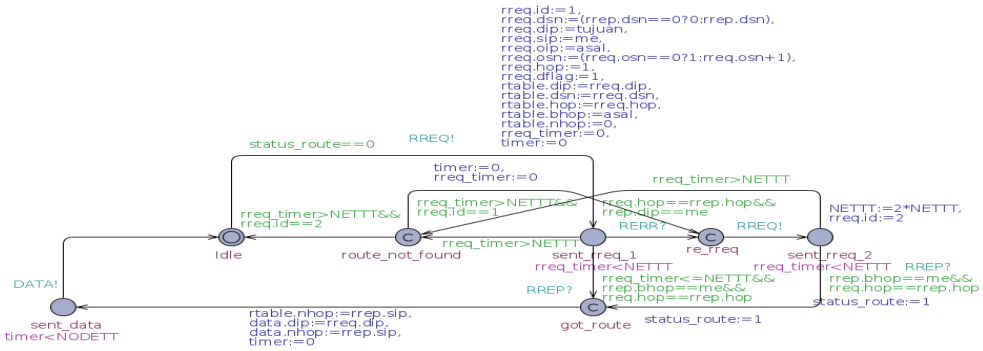
$$\frac{Interi.process\_rreq \xrightarrow{rrep.signal\ \&rrep.hop,\ RREP!} Interi.Idle \wedge Pengirim.Idle \xrightarrow{rrep.bhop==me,\ \ rrep.hop==rreq.hop\ \ RREP!J} Pengirim.got\_route}{< Interi.process\_rreq,\ Pengirim.Idle > \xrightarrow{rrep.bhop==me,\ \ rrep.hop==rreq.hop,\ \ rrep.signal\ \&rrep.hop\ \ \alpha J} < Interi.Idle, Pengirim.got\_route >} \quad (10)$$

2.5.2 Protokol AODV dalam *timed automata*

Verifikasi dilakukan dengan memodelkan protokol menjadi tiga *timed automata* yang berjalan secara pengiriman RREQ sesuai dengan persamaan (5)-(7) dan pengiriman RREP seperti pada persamaan (8)-(10). Ketiga *timed automata* tersebut yaitu *node* pengirim, *node intermediate*, dan *node* penerima.

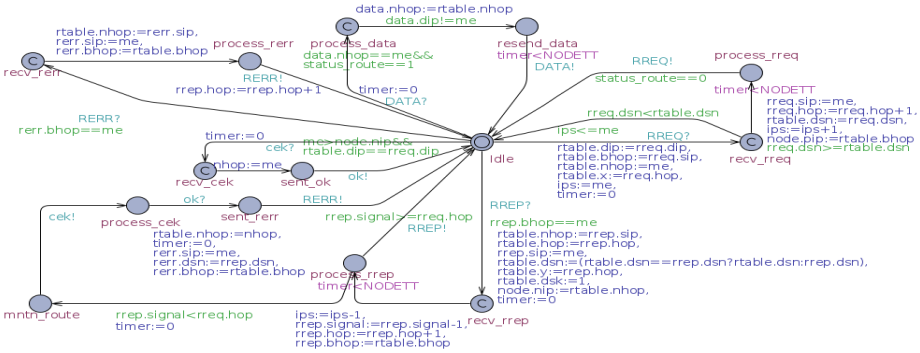


Gambar 2 Timed automata node pengirim AODV Reliable Delivery

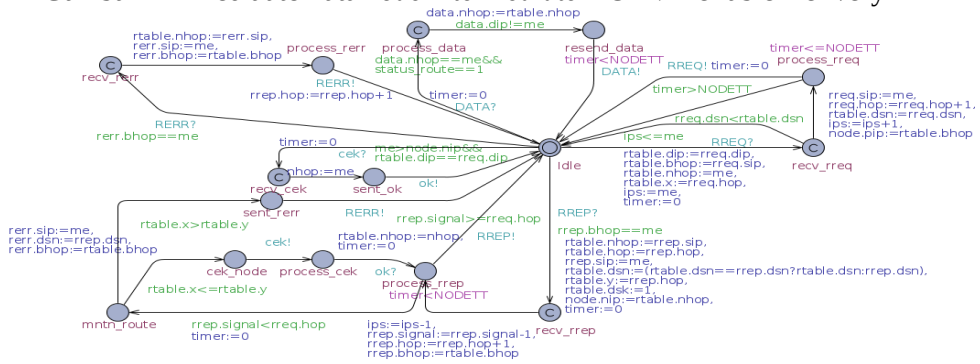


Gambar 3 Timed automata node pengirim AODV Break Avoidance

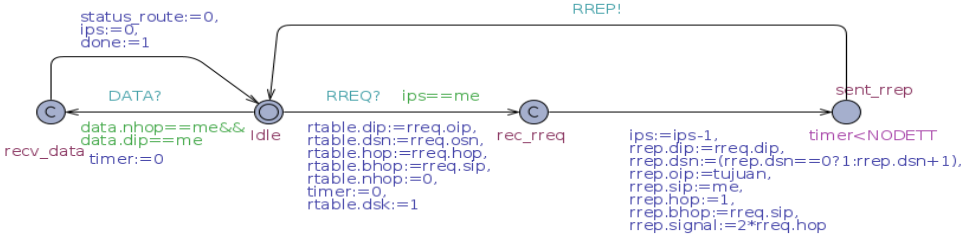
Timed Automata node pengirim protokol AODV RD ditunjukkan pada Gambar 2, dan protokol AODV BA pada Gambar 3. Adapun *timed automata node intermediate* pada Gambar 4 untuk protokol AODV RD dan Gambar 5 untuk protokol AODV BA. *Node* pengirim untuk protokol AODV RD pada Gambar 6 dan Gambar 7 untuk protokol AODV BA. *Node* pengirim adalah *node* yang menginisialisasi proses *route discovery* yaitu dengan mengirim RREQ dan mengirim paket. *Node* penerima adalah *node* yang menginisialisasi RREP sebagai tanda bahwa RREQ dari *node* asal telah diterima dan menerima paket dari *node* pengirim. RREP akan dikirim secara *unicast* melalui *node intermediate* yang sebelumnya telah menerima RREQ. *Node intermediate* adalah *node* perantara yang berada diantara *node* pengirim dan *node* penerima.



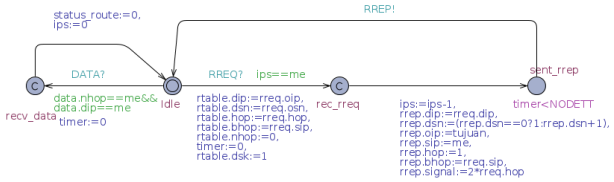
Gambar 4 Timed automata *node intermediate* AODV Reliable Delivery



Gambar 5 Timed automata *node intermediate* AODV Break Avoidance



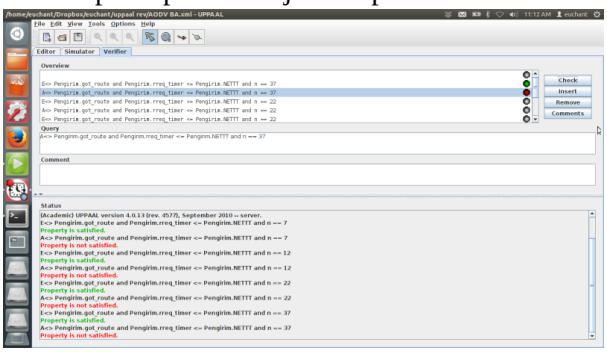
Gambar 6 Timed automata *node* penerima AODV Reliable Delivery



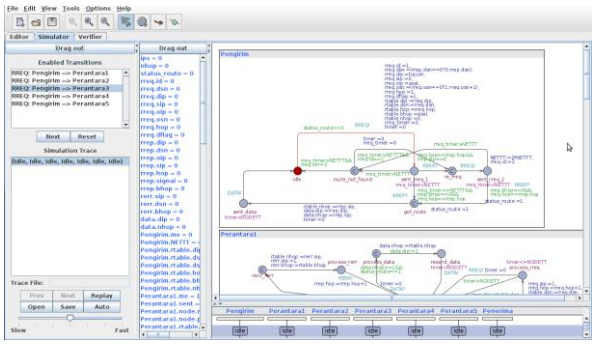
Gambar 7 Timed automata *node* penerima AODV Break Avoidance

3. HASIL DAN PEMBAHASAN

Verifikasi dari model protokol dilakukan dengan menggunakan tool model checker UPPAAL. Hasil verifikasi dari UPPAAL menunjukkan bahwa protokol AODV BA mampu memenuhi properti *route discovery* untuk persamaan (1) seperti ditunjukkan pada Gambar 8. Untuk persamaan (2), hasil dari verifikasi menunjukkan bahwa *timed automata* tidak mampu memenuhi properti *route discovery* dengan *diagnostic trace* yang diberikan sama untuk keadaan *trace* terpendek dan *trace* tercepat seperti ditunjukkan pada Gambar 9.

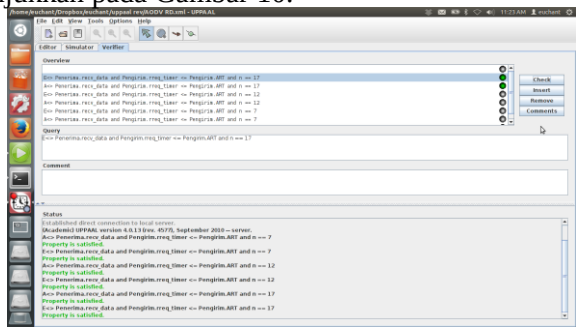


Gambar 8 Hasil verifikasi AODV Break Avoidance



Gambar 9 *Diagnostic trace* AODV Break Avoidance

Adapun hasil verifikasi untuk properti *packet delivery* pada model protokol AODV RD, UPPAAL menunjukkan bahwa model protokol mampu memenuhi properti ini untuk persamaan (3) dan (4) seperti ditunjukkan pada Gambar 10.



Gambar 10 Hasil verifikasi AODV Reliable Delivery

4. KESIMPULAN

Berdasarkan penelitian dan pembahasan yang telah dilakukan maka diperoleh kesimpulan bahwa verifikasi protokol AODV Reliable Delivery dan AODV Break Avoidance dilakukan dengan memodelkan protokol menjadi tiga *timed automata* dan diverifikasi secara otomatis menggunakan *tool model checker* UPPAAL. Berdasarkan hasil verifikasi, AODV Break Avoidance mampu memenuhi properti *route discovery* dengan *quantifier* ∃ akan tetapi tidak dapat memenuhi properti *route dicovery* dengan *quantifier* ∀ . Adapun protokol AODV Reliable Delivery mampu memenuhi properti *packet delivery* untuk kedua *quantifier* ∃ dan ∀ .

5. SARAN

Pada penelitian ini pemodelan *timed automata* untuk protokol AODV Break Avoidance dan AODV Reliable Delivery yang dilakukan hanya terbatas pada satu *node* pengirim, satu *node* penerima dan n *node intermediate.* Sehingga pada penelitian yang akan datang diharapkan mampu dilakukan verifikasi dengan jumlah *node* pengirim dan *node* penerima dapat berjumlah lebih dari satu.

Pembahasan terhadap penggunaan *timed automata* untuk memodelkan dan memverifikasi sebuah protokol diharapkan dapat digunakan untuk penelitian pada protokol-protokol lainnya, khususnya protokol yang memiliki ketergantungan terhadap aspek waktu tertentu berdasarkan spesifikasinya.

DAFTAR PUSTAKA

[1] Perkins, C.E., Das, S.R., and Belding-Royer, E.M., 2003, Ad hoc On-demand Distance Vector (AODV). Request for Comments (Experimental) RFC 3561, IETF.

[2] Wibling, O., Parrow, J. and Pears, A., 2004, Automatized Verification of Ad Hoc Routing Protocols. *Proceedings of International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, Madrid, 27 – 30 September.

[3] Jian, L. and Fang-min L, 2009, An Improvement of AODV Protocol Based on Reliable Delivery in Mobile Ad hoc Networks, *Proceedings of 5th IEEE International Conference on Information Assurance and Security (IAS)*, China, 18 – 20 Agustus.

[4] Khosrozadeh, A., Akbari, A., Bagheri, M. and Beikmahdavi, N., 2011, New AODV Routing Protocol with Break Avoidance, *Proceedings of IEEE International Symposium on Computer Science and Society (ISCCS)*, Malaysia, 16 Juli.

[5] Fehnker, A., van Glabbeek, R.J., Hofner, P., McIver, A., Portmann, M. and Tan, W.L., 2013, A Process Algebra for Wireless Mesh Networks Used for Modelling, Verifying and Analysing AODV, Technical Report 5513,National ICT Australia (NICTA), Sidney.

[6] Chiyangwa, S. and Kwiatkowska, M., 2005, A Timing Analysis of AODV, *Proceedings of 7th International Conference Formal Methods for Open Object-Based Distributed Systems*, Athena, 15 – 17 Juni.

[7] Baier, C. and Katoen, J.P., 2008, *Principles of Model Checking*, The MIT Press, Cambridge.