ANALISIS KINERJA PROTOKOL *ROUTING AD HOC ON-DEMAND DISTANCE VECTOR* (AODV) PADA JARINGAN *AD HOC HYBRID*: PERBANDINGAN HASIL SIMULASI DENGAN NS-2 DAN IMPLEMENTASI PADA *TESTBED* DENGAN PDA

Riri Fitri Sari, Abdusy Syarif, dan Bagio Budiardjo

Departemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia, Depok 16424, Indonesia

E-mail: riri@eng.ui.ac.id, abdusyarif@yahoo.com, bbudi@ee.ui.ac.id

Abstrak

Pada *Mobile Ad hoc NETwork (MANET)*, *node* yang dilengkapi dengan peralatan *wireless* memiliki kemampuan untuk mengelola dan mengorganisasi secara mandiri, walaupun tanpa kehadiran suatu infrastruktur jaringan. Jaringan *ad hoc hybrid*, memungkinkan beberapa *node* yang bergerak bebas (*mobile*) membangun komunikasi yang seketika (*instant*) dan terbebas dari ketergantungan pada infrastruktur dapat mengakses ke *Local Area Network* (LAN) atau ke Internet. Fungsi dari jaringan *ad hoc* sangat tergantung pada *routing protocol* yang menentukan jalur atau rute diantara *node*. *Ad hoc On-demand Distance Vector* (AODV) adalah salah satu *routing protocol* pada jaringan *ad hoc* yang bersifat *reactive.* Protokol ini adalah salah satu protokol yang paling banyak diteliti dan digunakan. Pada penelitian ini dilakukan pengkajian protokol AODV dengan membangun suatu *testbed* menggunakan *Personal Computer*, beberapa Laptop (sistem operasi Linux Red Hat 9.0 dan Fedora Core 2), serta *Personal Digital Assitant* (PDA). Penelitian ini juga membuat *package* yang lengkap dengan cara *cross compilation* untuk PDA iPAQ. Hasil yang didapat dari analisa simulasi protokol AODV dengan menggunakan *Network Simulator* NS-2 didapatkan rata-rata *packet delivery ratio* 99,89% , *end-to-end delay* sebesar 0,14 detik dan *routing overhead* sebesar 1.756,61 byte per detik. Kemudian hasil pengukuran simulasi dibandingkan dengan hasil pengukuran *testbed*. Dari hasil pengukuran *testbed* didapatkan *packet delivery ratio* adalah sebesar 99,57%, *end-to-end delay* sebesar 1,004 detik dan *routing overhead* sebesar 1.360,36 byte per detik.

Abstract

Ad Hoc on-Demand Distance Vector (AODV) Routing Protocol Performance Evaluation on Hybrid Ad Hoc Network: Comparison of Result of Ns-2 Simulation and Implementation on Testbed using PDA. In Mobile Ad hoc NETwork (MANET), node supplemented with wireless equipment has the capacity to manage and organise autonomously, without the presence of network infrastructures. Hybrid ad hoc network, enable several nodes to move freely (mobile) to create instant communication. Independent from infrastructure. They could access the Local Area Network (LAN) or the Internet. Functionalities of ad hoc network very much dependent on the routing protocol that determines the routing around node. Ad hoc On-demand Distance Vector (AODV) is one of routing protocols in ad hoc network which has a reactive characteristic. This protocol is the most common protocol being researched and used. In this Research, AODV protocol investigation was conducted by developing a testbed using Personal Computer, several Laptops (the Linux Red Hat operation system 9.0 and Fedora Core 2), and Personal Digital Assistant (PDA). This research also made a complete package by mean of cross compilation for PDA iPAQ. In general, results obtained from the simulation of AODV protocol using Network Simulator NS-2 are packet delivery ratio 99.89%, end-to-end delay of 0.14 seconds and routing overhead of 1,756.61 byte per second. Afterwards results from simulation were compared to results from testbed. Results obtained from testbed are as follows: the packet delivery ratio is 99.57%, the end-to-end delay is 1.004 seconds and the routing overhead is 1,360.36 byte per second.

Keywords: MANET, ad hoc hybrid, protocol routing AODV

1. Pendahuluan

Sejalan dengan perkembangan teknologi informasi yang terus berkembang, diperlukan suatu jenis atau tipe jaringan khusus yang mampu melibatkan banyak orang atau peralatan komunikasi tanpa ketergantungan terhadap suatu infrastruktur. Ini yang dimaksud dengan *Mobile Ad hoc NETwork (MANET).*

MANET bisa terbentuk dari sekumpulan *node* yang menggunakan antarmuka nirkabel (*wireless interface*) mereka untuk melakukan komunikasi antara satu *node* dengan *node* yang lainnya. Setiap *node* bisa menjadi *host* ataupun *router*, sehingga *node* mampu meneruskan paket ke *node* berikutnya. Selanjutnya, agar *node* bisa berkomunikasi dengan *node* yang diluar jangkauannya, diperlukan protokol *routing* yang memiliki kemampuan untuk melewati banyak titik / *node* (*multihop*), sehingga MANET juga diharapkan menjadi jaringan dengan jangkauan yang lebih luas dibanding jaringan radio.

Protokol *routing* pada jaringan *ad hoc* menjadi suatu permasalahan yang menantang untuk diteliti semenjak sebuah *node* bisa bergerak secara bebas (acak). Pada *ad hoc* ada dua tipe protokol *routing* [1], yaitu :

- Proaktif: *Destination Sequenced Distance Vector* (DSDV), *Cluster Switch Gateway Routing* (CSGR), *Wireless Routing Protocol* (WRP), *Optimized Linkstate* (OLSR)
- Reaktif: *Dynamic Source Routing* (DSR), *Ad hoc On-demand Distance Vector* (AODV), *Temporally Ordered Routing Algorithm* (TORA), Associativy Based Routing (ABR), *Signal Stability Routing* (SSR)

Ada juga pendekatan pada protokol *routing hybrid* yang mengkombinasikan antara kedua tipe protokol routing, proaktif dan reaktif, contohnya *Zone Routing Protocol* (ZRP).

AODV adalah salah satu protokol *routing* yang paling banyak diteliti pada lingkungan ad hoc. Bjorn Wiberg dan Erik Nordstrom dari Uppsala University, Swedia, pada akhir tahun 2002 telah berhasil mengembangkan protokol *routing* berbasis AODV dan diberi nama AODV-UU. Protokol ini dapat digunakan baik pada simulasi (menggunakan ns-2.1b9a pada Linux Red Hat 7.2) maupun pada lingkungan sesungguhnya (pada percobaannya hanya menggunakan 3 buah laptop). Sampai laporan ini ditulis, versi AODV-UU sudah mencapai versi 0.9. Pada [2], Alex Hamidian diawal tahun 2003 telah mengembangkan protokol AODV+ yang menyediakan koneksi antar jaringan (*network interconnection)*, dan semua komunikasi antara *mobile ad hoc network* dan Internet harus melalui sebuah *gateway*. Tetapi AODV+ ini masih sebatas simulasi dan masih terus dikembangkan. Selain itu ada AODV-UCSB yang dikembangkan oleh University of California-Santa Barbara, dan ternyata AODV-UCSB ini menggunakan modul dari AODV-UU. Dan AODV-UCSB ini memiliki 2 versi untuk sistem operasi yang berbeda, yaitu Linux dan Microsoft (hanya Microsoft Windows XP). Selanjutnya ada *Ad hoc On-demand Multipath Distance Vector* (AOMDV) yang mengembangkan AODV untuk *multipath* oleh Samir R. Das dan Mahesh K. Marina dari *Departement of Computer Science,* State University of New York. Proyek lainnya adalah *MobileMAN* [3], yang merupakan proyek dari Jose Costa-Requena dan kawan-kawan [4] dari Helsinky University of Technology, Finlandia, yang juga telah melakukan implementasi jaringan *ad hoc* (*testbed* dengan menggunakan 6 buah PDA iPAQ h3900) berbasis protokol AODV-UU. Sekarang Jose sedang mencoba memanfaatkan *Session Initiate Protocol* (SIP) pada jaringan ad hoc. Dari hasil komunikasi dengan peneliti dari Helsinky University of Technology didapat informasi bahwa mereka telah berhasil membuat *ad hoc backbone*.

2. Metode Penelitian

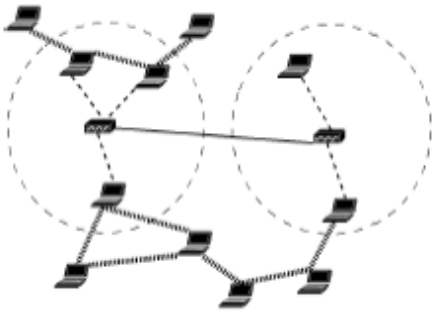
Jaringan Hybrid

Konsep jaringan *hybrid* adalah gabungan dari jaringan infrastruktur dan *ad hoc* atau MANET. Disatu sisi ada jaringan dengan infrastruktur dan disisi lain terdapat *node* yang dapat bergerak bebas (*mobile node*) dengan fasilitas routing. Hal ini memungkinkan rute *multi-hop* antara *mobile node* dengan *base station.* Hal ini menyebabkan ruang lingkup area dari *base station* menjadi lebih luas, seperti terlihat pada Gambar 1. Ide jaringan *hybrid* adalah untuk efisiensi dari jaringan infrastruktur yang ada, untuk memperluas area dari *base station* dan mengurangi konsumsi daya listrik.

***Ad hoc On-demand Distance Vector* (AODV).** AODV adalah *distance vector routing protocol* yang termasuk dalam klasifikasi reaktif *routing protocol*, yang hanya me-*request* sebuah rute saat dibutuhkan. AODV yang standar ini dikembangkankan oleh C. E. Perkins, E.M. Belding-Royer dan S. Das pada RFC 3561 [5].

Ciri utama dari AODV adalah menjaga *timer-based state* pada setiap *node* sesuai dengan penggunaan tabel *routing*. Tabel r*outing* akan kadaluarsa jika jarang digunakan.

AODV memiliki *route discovery* dan *route maintenance*. *Route Discovery* berupa *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *Route Maintenance* berupa Data, *Route update* dan *Route Error* (RRER).



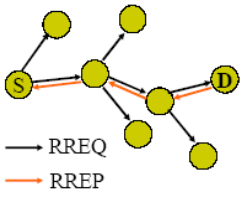
Gambar 1. Jaringan *Hybrid* [6]

AODV memerlukan setiap *node* untuk menjaga tabel *routing* yang berisi *field* :

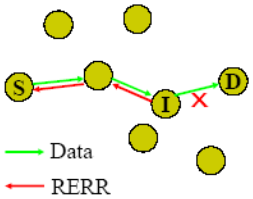
- *Destination IP Address*: berisi alamat IP dari *node* tujuan yang digunakan untuk menentukan rute.
- *Destination Sequence Number* : *destination sequence number* bekerjasama untuk menentukan rute
- *Next Hop*: 'Loncatan' (*hop*) berikutnya, bisa berupa tujuan atau *node* tengah, *field* ini dirancang untuk meneruskan paket ke *node* tujuan.
- *Hop Count*: Jumlah *hop* dari alamat IP sumber sampai ke alamat IP tujuan.
- *Lifetime*: Waktu dalam milidetik yang digunakan untuk *node* menerima RREP.
- *Routing Flags*: Status sebuah rute; *up* (valid), *down* (tidak valid) atau sedang diperbaiki.

AODV mengadopsi mekanisme yang sangat berbeda untuk menjaga informasi *routing*. AODV menggunakan tabel *routing* dengan satu *entry* untuk setiap tujuan. Tanpa menggunakan *routing* sumber, AODV mempercayakan pada tabel *routing* untuk menyebarkan *RouteReply* (RREP) kembali ke sumber dan secara sekuensial akan mengarahkan paket data menuju ketujuan. AODV juga menggunakan *sequence number* untuk menjaga setiap tujuan agar didapat informasi *routing* yang terbaru dan untuk menghindari *routing loops*. Semua paket yang diarahkan membawa *sequence number* ini.

Penemuan jalur (*Path discovery*) atau *Route discovery* di-inisiasi dengan menyebarkan *RouteReply* (RREP), seperti terlihat pada Gambar 2. Ketika RREP menjelajahi *node*, ia akan secara otomatis men-setup *path*. Jika sebuah *node* menerima RREP, maka *node* tersebut akan mengirimkan RREP lagi ke *node* atau *destination sequence number*. Pada proses ini, *node* pertama kali akan mengecek *destination sequence*



Gambar 2. Mekanisme Penemuan Rute [1]



Gambar 3. Mekanisme Data (*Route Update*) dan *Route Error* [1]

number pada tabel *routing*, apakah lebih besar dari 1 (satu) pada *RouteRequest* (RREQ), jika benar, maka *node* akan mengirim RREP. Ketika RREP berjalan kembali ke source melalui *path* yang telah di-setup, ia akan men-setup jalur kedepan dan meng-*update timeout*.

Jika sebuah *link* ke *hop* berikutnya tidak dapat dideteksi dengan metode penemuan rute, maka link tersebut akan diasumsikan putus dan *RouteError* (RERR) akan disebarkan ke *node* tetangganya seperti terlihat pada Gambar 3. Dengan demikian sebuah *node* bisa menghentikan pengiriman data melalui rute ini atau meminta rute baru dengan menyebarkan RREQ kembali.

Network Simulator

Network Simulator NS-2 [7] adalah suatu *interpreter* yang *object-oriented*, dan *discrete event-driven* yang dikembangkan oleh *University of California Berkeley* dan USC ISI sebagai bagian dari projek *Virtual INternet Testbed* (VINT). NS menjadi salah satu *tool* yang sangat berguna untuk menunjukkan simulasi jaringan melibatkan *Local Area Network* (LAN), *Wide Area Network* (WAN), tapi fungsi dari *tool* ini telah berkembang selama beberapa tahun belakangan ini untuk memasukkan didalamnya jaringan nirkabel (*wireless*) dan juga jaringan ad hoc.

Network Simulator pertama kali dibangun sebagai varian dari *REAL Network Simulator* pada tahun 1989 di *University of California Berkeley* (UCB). Pada tahun 1995 pembangunan *Network Simulator* didukung oleh *Defense Advanced Research Project Agency* (DARPA) melalui VINT Project, yaitu sebuah tim riset gabungan yang beranggotakan tenaga-tenaga ahli dari beberapa instansi ternama [8].

Ada beberapa keuntungan menggunakan NS sebagai perangkat lunak simulasi pembantu analisi dalam riset, antara lain adalah NS dilengkapi dengan *tool* validasi yang digunakan untuk menguji kebenaran pemodelan yang ada pada NS. Secara default, semua pemodelan NS akan dapat melewati proses validasi ini. Pemodelan media, protokol dan komponen jaringan yang lengkap dengan perilaku trafiknya sudah disediakan pada *library* NS.

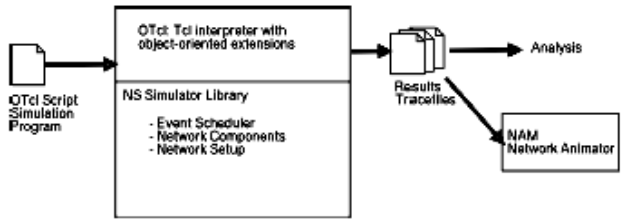
NS juga bersifat *open source* dibawah *Gnu Public License* (GPL), sehingga NS dapat di-*download* dan digunakan secara gratis melalui *web site* NS yaitu http://www.isi.edu/nsnam/. Sifat *open source* juga mengakibatkan pengembangan NS menjadi lebih dinamis.

Struktur NS. Ns dibangun menggunakan metoda *object oriented* dengan bahasa C++ dan OTcl (*variant object oriented* dari Tcl). Seperti terlihat pada Gambar 4,

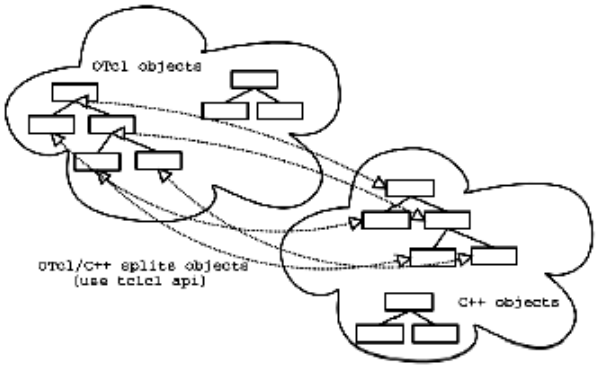
ns-2 menginterpretasikan *script* simulasi yang ditulis dengan OTcl. Seorang user harus mengeset komponen-komponen (seperti objek penjadwalan *event, library* komponen jaringan, dan *library* modul *setup*) pada lingkungan simulasi.

User menuliskan simulasinya dengan *script* OTcl, dan menggunakan komponen jaringan untuk melengkapi simulasinya. Jika user memerlukan komponen jaringan baru, maka user dengan bebas untuk menambahkan dan mengintegrasikan pada simulasinya atau pada ns-2. penjadwalan *event* (*event scheduler*) berfungsi sebagai komponen utama selain pencetus (*trigger) event* komponen jaringan simulasi (seperti mengirim paket, memulai dan menghentikan *tracing*). Sebagian dari ns-2 ditulis dalam Bahasa C++ untuk alasan efisiensi. Jalur data (*data path*), ditulis dalam Bahasa C++, dipisahkan dari jalur kontrol (*control path*), ditulis dalam Bahasa OTcl. Objek jalur data dikompilasi dan kemudian interpreter OTcl melalui OTcl *linkage* (tclcl) yang memetakan metode dan variabel pada C++ menjadi objek dan variabel pada OTcl. Objek C++ dikontrol oleh objek OTcl. Hal ini memungkinkan untuk menambahkan metode dan variabel kepada C++ yang dihubungkan dengan objek OTcl. Hirarki *linked class* pada C++ memiliki korespondansi dengan OTcl, hal ini dapat dilihat pada Gambar 5.

Hasil yang dikeluarkan oleh ns-2 berupa file *trace*, harus diproses dengan menggunakan *tool* lain, seperti *Network Animator* (NAM), *perl*, *awk*, atau *gnuplot* [9].



Gambar 4. Skema NS [6]



Gambar 5. C++ dan OTcl : Duality [6]

Fungsi NS. Beberapa fungsi yang tersedia pada ns-2 adalah untuk jaringan dengan kabel atau tanpa kabel, *tracing*, dan visualisasi. Lebih jelasnya:

- Mendukung jaringan dengan kabel
  - Protokol routing *Distance Vector, Link State*
  - Protokol Transport : TCP, UDP
  - Sumber trafik : web, ftp, telnet, cbr, real audio
  - Tipe antrian yang berbeda : drop tail, RED
  - *Quality of Service* (QoS) : *Integrated Services* dan *Differentiated Services*
  - *Emulation*
- Mendukung jaringan tanpa kabel (*wireless*)
  - Protokol routing ad hoc: AODV, DSR, DSDV, TORA; Jaringan *hybrid*; *Mobile IP*; Satelit; Senso-MAC; Model propagasi: *two-ray ground, free space, shadowing*
- *Tracing*
- Visualisasi
  - *Network Animator* (NAM)
  - *TraceGraph*
- Kegunaan
  - Pembangkit pergerakan *mobile*
    setdest –v (versi) –n (jumlah *node*) –p (waktu pause) –s (kecepatan) –t (waktu simulasi) –x (panjang area) –y (lebar area) > (File keluaran)
  - Pembangkit pola trafik (CBR / TCP *traffic*)
    Ns cbrgen.tcl [-type cbr | tcp] [-nn jumlah *node*] [-seed seed] [-mc koneksi] [-rate rata-rata]

Table 1. Format File Trace [10]

Event	Abbreviation	Flag	Type	Value
Wireless Event	s: Send r: Receive d: Drop f: Forward	-t	double	Time (* For Global Setting)
		-Ni	int	*Node* ID
		-Nx	double	*Node* X Coordinate
		-Ny	double	*Node* Y Coordinate
		-Nz	double	*Node* Z Coordinate
		-Ne	double	*Node* Energy Level
		-Nl	string	Network trace Level (AGT, RTR, MAC, etc.)
		-Nw	string	Drop Reason
		-Hs	int	Hop source *node* ID
		-Hd	int	Hop destination *Node* ID, -1, -2
		-Ma	hexadecimal	Duration
		-Ms	hexadecimal	Source Ethernet Address
		-Md	hexadecimal	Destination Ethernet Address
		-Mt	hexadecimal	Ethernet Type
		-P	string	Packet Type (arp, dsr, imep, tora, etc.)
		-Pn	string	Packet Type (cbr, tcp)

Format File *Trace*. Terdapat dua tipe format file *trace*, lama dan baru. Setiap *trace* dimulai dengan sebuah karakter atau singkatan yang menandakan tipe *trace*, selanjutnya diikuti dengan format *trace* yang tetap atau variabel.

Format file trace dimulai dengan satu sampai empat karakter (seperti terlihat pada Tabel 1). Kemudian diikuti dengan *flag* atau sepasang nilai yang sama dengan *trace* NAM. Huruf pertama dari *flag* dengan dua huruf menunjukkan tipe *flag :*

- N : Sifat *Node*
- I : Informasi tingkat paket IP
- H : Informasi 'loncatan' berikutnya (*Next Hop*)
- M : Informasi tingkat paket MAC
- P : Informasi spesifik paket

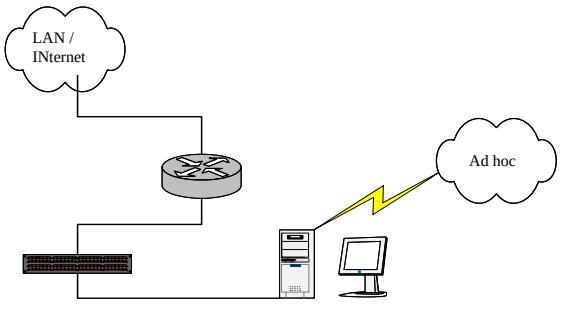
Nilai untuk *flag* –Hd bisa –1 atau –2. Jika bernilai –1, maka artinya paket itu adalah paket *broadcast*, dan jika bernilai –2, maka artinya *node* tujuan belum ditentukan. Nilai –2 umumnya digunakan untuk paket yang melalui tingkatan *agent* (-N1 AGT) dan *routing* (-N1 RTR).

RancanganTestbed

Berikut ini adalah perangkat keras yang digunakan untuk lingkungan percobaan :

- Laptop, dengan spesifikasi : Intel Pentium IV, 2400 MHz, RAM 256 MB, HardDisk 30 GB, PCMCIA Wireless Card Cisco Aironet 350 series
- PC desktop, dengan spesifikasi Intel Pentium IV, 1700 MHz, RAM 128 MB, *HardDisk* 20 GB, PCI *Wireless* D-Link DWL-G520
- PDA, Hewlett Packard iPAQ seri 5550 dengan spesifikasi Intel XScale 400 MHz, RAM 128 MB, ROM 48 MB, *Wireless*, *Infra red*, *Bluetooth* (*integrated build-in)*

Berikut ini perangkat lunak yang digunakan untuk lingkungan percobaan adalah Sistem operasi Linux (Red Hat 9.0, Fedora Core 2), Sistem operasi Linux Familiar v0.7.2 (kernel 2.4.19-rmk6-pxa1-hh36), AODV-UU[11], Monmotha *Cross Compiler* [12], Ethereal [13], Iperf [14], Iptables [15].



Gambar 6. Kedudukan *Ad hoc Gateway*

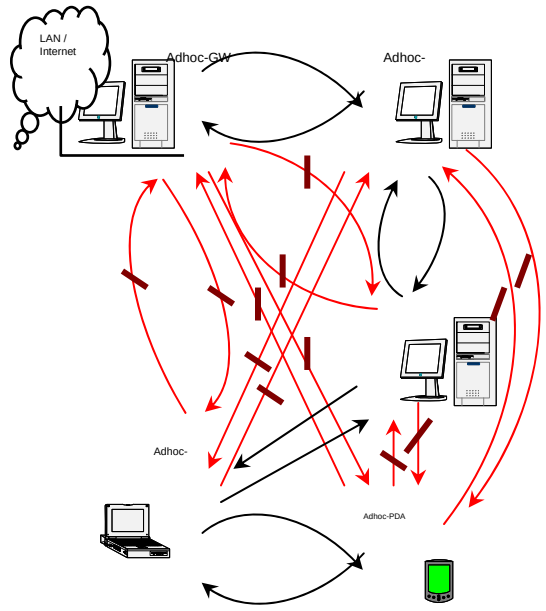
Gateway. *Gateway* pada penelitian ini dibuat dari sebuah PC yang dilengkapi dengan 2 buah kartu jaringan (*PCI Card*), eth0 dan wlan0. Eth0 adalah peralatan yang terhubung ke jaringan infrastruktur menggunakan kabel, dan wlan0 digunakan sebagai alat komunikasi pada jaringan *ad hoc* seperti lihat Gambar 6.

Dibuat juga *script* untuk konfigurasi *gateway* pada jaringan *ad hoc hybrid*. Tujuan dari pembuatan *script* ini adalah agar semua paket data dari *node* yang berada pada jaringan *ad hoc* bisa diteruskan ke jaringan infrastruktur (LAN atau Internet) dan sebaliknya.

Iptables berfungsi sebagai penyaring paket (*packet filtering*). Struktur perintah yang digunakan untuk membuang (*drop*) paket yang datang dari suatu *node* adalah seperti berikut :
"iptables –A INPUT –m mac –-mac-source ab:cd:ef:gh:ij:kl –j DROP"

dimana ab:cd:ef:gh:ij:kl adalah *MAC address* suatu *node* yang semua pesan (*message*) harus dibuang (*drop*). Semua *MAC address* setiap *Network Interface Card* (NIC) harus dicatat untuk menentukan *MAC address* dari suatu *node* yang harus dibuang.

Skenario pada Gambar 7 akan memaksa sebuah *node* hanya boleh melewati *node* yang telah ditentukan. Hal ini dapat dikerjakan oleh *iptables* dengan perintah seperti contoh diatas. Untuk lebih jelasnya perhatikan Tabel 2. yang memperlihatkan *MAC address* dari suatu *node* yang harus dibuang (*drop*) pada masing-masing *node*. Contoh, *node* Adhoc-GW membuang semua paket yang berasal dari *MAC address node* Adhoc-PC2, Adhoc-NB dan Adhoc-PDA, sehingga *node-node* tersebut tidak dapat berhubungan secara langsung dengan 1 hop.



Gambar 7. Skenario Multihop

Tabel 2. Konfigurasi *Drop MAC Address*

Node	***Node yang dibuang (*Drop*)* MAC Address**
Adhoc-GW	Adhoc-PC2, Adhoc-NB Adhoc-PDA
Adhoc-PC1	Adhoc-NB, Adhoc-PDA
Adhoc-PC2	Adhoc-GW, Adhoc-PDA
Adhoc-NB	Adhoc-GW, Adhoc-PC1
Adhoc-PDA	Adhoc-GW, Adhoc-PC1 Adhoc-PC2

Sedangkan perintah untuk menerima semua paket yang masuk melalui wlan0 kemudian diteruskan (*POSTROUTING*) melalui eth0 dari semua *IP address* untuk tujuan LAN atau Internet adalah sebagai berikut : "iptables –t nat -A POSTROUTING -s 0/0 -0 eth0 -j MASQUERADE"

Dengan menjalankan *script* diatas akan memperbolehkan semua *node* pada jaringan *ad hoc* mengakses LAN atau Internet.

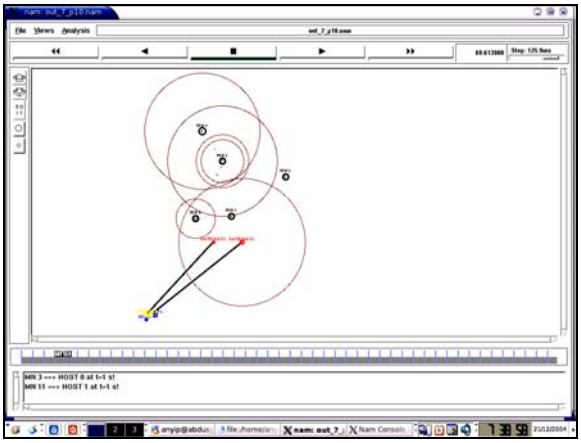
3. Hasil dan Pembahasan

Beberapa parameter dievaluasi dan dianalisa berdasarkan simulasi yang dilakukan. Adapun parameter yang diukur adalah:

- Perbandingan paket terkirim (*packet delivery ratio*), adalah jumlah paket data yang diterima dibagi dengan jumlah paket data yang dibangkitkan.
- *End-to-end delay*, waktu kedatangan paket data pada destination dikurangi waktu paket data yang dibangkitkan oleh *source*.
- *Routing Overhead*, adalah jumlah pesan AODV yang ditransmisikan.

Skenario Simulasi. Untuk mengevaluasi dan menganalisa kinerja dari protokol *routing* AODV pada jaringan *ad hoc hybrid*, dilakukan sebuah simulasi dan dilakukan beberapa kali simulasi. Pada bagian ini akan dijelaskan mengenai simulasi, skenario dan parameter yang digunakan dan kinerja yang akan diukur.

Simulasi ini dilakukan pada sebuah laptop Intel Pentium IV, 2400 MHz, RAM 128 MB, dan dengan sistem operasi Linux Mandrake 9.2. Skenario simulasi seperti terlihat pada Gambar 8 yang terdiri dari 5 buah *node*, 2 *gateway*, 2 router dan 2 host. Kelima *node* tersebut menggunakan tipe data *Transmission Control Protocol* (TCP). Sedangkan topologi simulasi ini adalah 500 x 400 meter berbentuk persegi panjang. Dua buah *gateway* diletakkan pada dua titik yang berbeda, menggunakan koordinat x,y dalam satuan meter, untuk *node* yang satu pada posisi titik koordinat (200,300) dan satunya lagi pada titik koordinat (300,300). Seluruh simulasi ini berlangsung selama 900 detik.



Gambar 8. Skenario Simulasi

Tabel 3. Parameter pada Simulasi

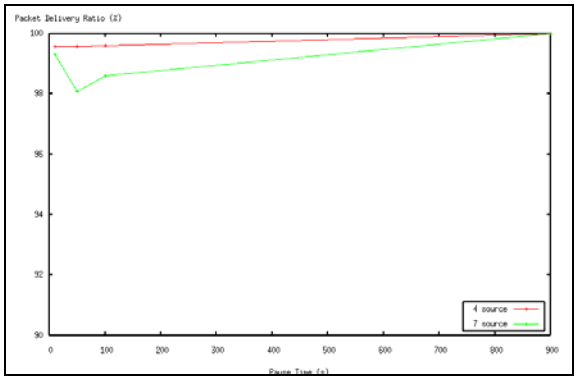
Parameter	Nilai
Jangkauan Transmisi	250 m
Waktu Simulasi	900 dtk
Ukuran Topologi	500 x 400 m
Jumlah *node Ad hoc*	5
Jumlah source	4
Jumlah *gateway*	2
Tipe Trafik	TCP
Packet rate	5 paket/ dtk
Packet size	64 bytes
Pause time	10 dtk
Maximum speed	5 m/dtk

Waktu yang digunakan untuk memulai pengiriman paket data adalah sepuluh detik setelah simulasi berlangsung. Setelah itu, kelima *Mobile node* tersebut secara terus menerus mengirimkan paket data sampai satu detik sebelum simulasi berhenti. *Node* yang dituju oleh setiap *node* adalah salah satu dari dua host, yang dipilih secara acak.

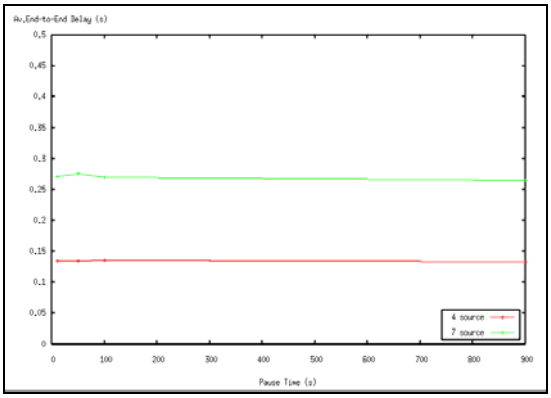
Pada simulasi ini, dipilih nilai untuk *pause time* dan kecepatan minimum juga maksimum dari *Mobile node* tersebut.

Parameter

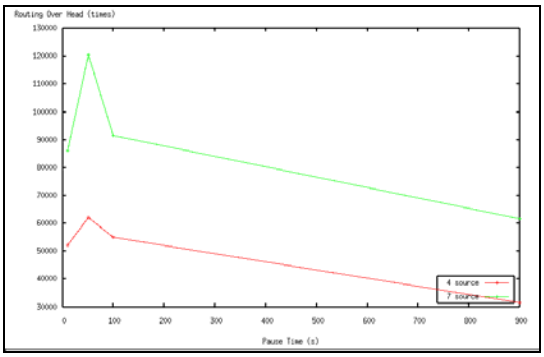
Beberapa parameter yang digunakan pada simulasi ini bisa dilihat pada Tabel 3. Jangkauan transmisi adalah jarak maksimum yang masih memungkinkan, antara dua *node*, yaitu 250 meter. Jika sebuah *node* bergerak menjauhi *node* lebih dari 250 meter atau diluar jangkauan transmisi, maka *node* tersebut disimpulkan tidak dapat berkomunikasi dengan *node* yang lain.



Gambar 9. Perbandingan Paket Terkirim Pada Simulasi



Gambar 10. Rata-rata *End-to-end Delay* Pada Simulasi



Gambar 11. Rata-rata *Routing Overhead* Pada Simulasi

Hasil Pengukuran Simulasi

Gambar 9 memperlihatkan grafik perbandingan paket terkirim (*packet delivery ratio*). Hasil yang didapat dengan simulasi memperlihatkan rasio mendekati 99 % untuk 4 *source node*, begitu juga pada saat menggunakan 7 *source node.* Pada 100 detik pertama simulasi berjalan, terjadi penurunan. Hal ini dipengaruhi oleh rute tabel yang berubah, sehingga tabel rute juga berubah. Sehingga *node* harus menyebarkan kembali pesan HELLO, *route request* (RREQ) sampai didapatkan *route reply* (RREP). Gambar 10 menggambarkan rata-rata *end-to-end delay* dengan *pause time* 100 detik. Dapat dilihat bahwa *end-to-end delay* cukup stabil yaitu berkisar 0.14 detik (untuk 4 *source*) dan 0.27 detik (untuk 7 *source*).

Gambar 11 memperlihatkan rata-rata routing *overhead* dari AODV. Terlihat adanya bentuk kurva yang tiba-tiba menurun secara drastis. Hal ini dikarenakan pada awal simulasi *node* terus menyebarkan RREQ ke semua *node* sampai didapatkannya RREP, sehingga hal ini menyebabkan menurunnya *bandwidth*.

Hasil Testbed

Sebagian *testbed* dilakukan di dalam ruangan Gedung Pascasarjana Fakultas Teknik Universitas Indonesia, Depok. Posisi dari masing-masing *node* bisa dilihat pada Gambar 12. Skenario ini dipilih dengan maksud agar masing-masing *node* masih berada dalam ruang lingkup area atau jangkauan frekuensi radio (*wireless*) dan terdapat banyak halangan (berupa dinding dan kaca), dan dapat melakukan multihop. Percobaan ini dilakukan sebanyak 5 kali untuk tiap hop.

Seperti terlihat pada Gambar 12, dalam *testbed* digunakan 1 buah *node* (Adhoc-GW) sebagai *gateway*, yang berfungsi sebagai jembatan antara jaringan *ad hoc* dan jaringan infrastruktur. Juga terdapat 3 buah *node* (Adhoc-PC1, Adhoc-PC2, Adhoc-NB) yang statik dan 1 buah *node* (Adhoc-PDA) yang dapat dinamik.

Untuk percobaan 1 hop hanya digunakan Adhoc-PC1 dan Adhoc -GW. Ketika percobaan diganti menjadi 2 hop, Adhoc-PC2 diaktifkan. Konektifitas antara *node* dapat dicek menggunakan PING. Untuk menguji multihop maka Adhoc-NB dan Adhoc-PDA diaktifkan.

Seperti telah dikatakan sebelumnya, bahwa dalam melakukan percobaan ini, penulis menggunakan hanya satu *collision domain.* Untuk menghindari terjadinya tabrakan (*collision*) digunakan metode *Carrier Sense Multiple Access / Collision Avoidance* (CSMA/CA) yaitu dengan mematikan beberapa *node* pada layer *MAC address*.

Tabel 4 memperlihatkan perbandingan Ping sukses yang dilakukan dari setiap *node ad hoc* ke sebuah *node* yang berada pada LAN melalui Adhoc-GW (*gateway*). Di dalam ruangan laboratorium, bisa didapat kinerja AODV dengan baik. Untuk 1 dan 2 hop, walaupun terdapat paket yang hilang (*loss*) dan sedikit duplikasi paket, hal ini masih tergolong normal. Jumlah paket yang hilang dan paket duplikasi dipengaruhi oleh jumlah hop didalam rute. Pada Tabel 4 bisa dilihat, bahwa paket yang hilang akan bertambah bersamaan dengan bertambahnya jumlah hop. Hilangnya paket tersebut masih normal. Dengan data tersebut belum dapat ditarik kesimpulan hanya melihat paket yang hilang saja. Hilangnya paket bisa disebabkan oleh beberapa faktor, misal faktor perangkat keras, seperti

stabilitas *link* bisa berlainan antara 1 percobaan dengan percobaan lain.

Percobaan 1 Hop

Pada percobaan ini digunakan 2 *node*, yaitu *node* Adhoc-PC1 dan Adhoc-GW seperti terlihat pada Gambar 13. Adhoc-PC1 pertama mencoba koneksi dengan Adhoc-GW. PING digunakan untuk menguji konektifitas antara *node*, baik antara *ad hoc node* pada jaringan *ad hoc* maupun LAN. Pengujian lain yang dilakukan adalah Adhoc-PC1 mencoba koneksi ke Internet dan LAN melalui Adhoc-GW.

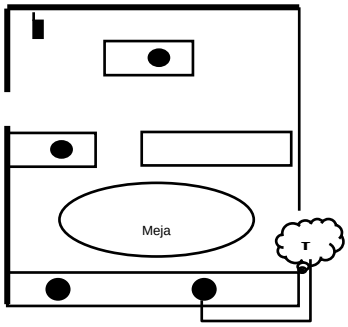
Gambar 13 memperlihatkan hasil pengukuran *bandwidth* yang ditangkap oleh *tools* Iperf. Rata-rata *bandwidth* yang didapat untuk 1 hop adalah 5,10 Mbps. Hal ini menunjukan kinerja AODV yang cukup baik.

Pada Gambar 15 memperlihatkan hasil pengujian koneksi antara *node* dalam percobaan 1 hop menggunakan PING yang secara terus menerus mengirim paket berukuran 64 bytes.

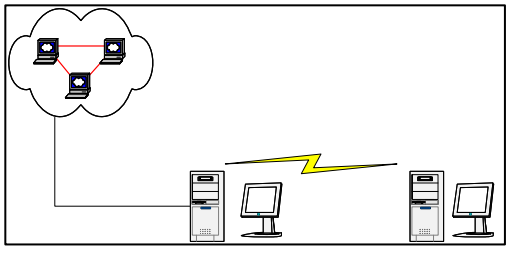
Terlihat tidak terdapat paket yang hilang (*packet loss*). Pengujian ini dilakukan sebanyak 5 kali. Hal ini juga berlaku untuk percobaan *multihop.*

Percobaan 2 Hop

Untuk melakukan percobaan 2 hop, *node* Adhoc-PC2 diaktifkan seperti terlihat pada Gambar 17 dan *node* menjalankan AODV-UU.



Gambar 12. Skenario *Testbed*



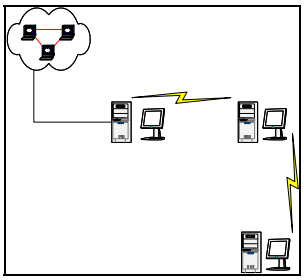
Gambar 14. Skenario Percobaan 1 Hop

```
[ 3] local 192.168.180.103 port 36215 connected
with 192.168.180.101 port 5001
[ 3]  0.0- 1.0 sec    672 KBytes  5.51 Mbits/sec
[ 3]  1.0- 2.0 sec    616 KBytes  5.05 Mbits/sec
[ 3]  2.0- 3.0 sec    616 KBytes  5.05 Mbits/sec
[ 3]  3.0- 4.0 sec    640 KBytes  5.24 Mbits/sec
[ 3]  4.0- 5.0 sec    624 KBytes  5.11 Mbits/sec
[ 3]  5.0- 6.0 sec    608 KBytes  4.98 Mbits/sec
[ 3]  0.0-10.0 sec   6.09 MBytes  5.10 Mbits/sec
  ID      Interval       Transfer      Bandwidth
```

Gambar 15. Rata-rata Bandwidth Pada percobaan 1 Hop

```
[root@Adhoc-PC1 root]# ping 192.168.180.101
PING 192.168.180.101 (192.168.180.101) 56(84) bytes of data.
64 bytes from 192.168.180.101: icmp_seq=1 ttl=64 time=1.16 ms
64 bytes from 192.168.180.101: icmp_seq=2 ttl=64 time=1.12 ms
64 bytes from 192.168.180.101: icmp_seq=3 ttl=64 time=1.12 ms
64 bytes from 192.168.180.101: icmp_seq=4 ttl=64 time=1.12 ms
64 bytes from 192.168.180.101
: icmp_seq=5 ttl=64 time=1.10 ms
64 bytes from 192.168.180.101: icmp_seq=6 ttl=64 time=1.17 ms
64 bytes from 192.168.180.101: icmp_seq=7 ttl=64 time=1.12 ms
64 bytes from 192.168.180.101: icmp_seq=8 ttl=64 time=1.13 ms
```

Gambar 16. *Output* PING 1 Hop



Gambar 17. Skenario Percobaan 2 Hop

```
[ 3]  0.0- 1.0 sec    304 KBytes  2.49 Mbits/sec
[ 3]  1.0- 2.0 sec    288 KBytes  2.36 Mbits/sec
[ 3]  2.0- 3.0 sec    288 KBytes  2.36 Mbits/sec
[ 3]  3.0- 4.0 sec    256 KBytes  2.10 Mbits/sec
[ 3]  4.0- 5.0 sec    304 KBytes  2.49 Mbits/sec
[ 3]  5.0- 6.0 sec    152 KBytes  1.25 Mbits/sec
[ 3]  6.0- 7.0 sec    296 KBytes  2.42 Mbits/sec
```

Gambar 18. Rata-rata *Bandwidth* Pada Percobaan 2 Hop

Gambar 18 memperlihatkan rata-rata *bandwidth* yang didapat adalah 2,15 Mbps. Ini menunjukkan terjadi penurunan *bandwidth* yang cukup besar jika dibandingkan dengan 1 hop.

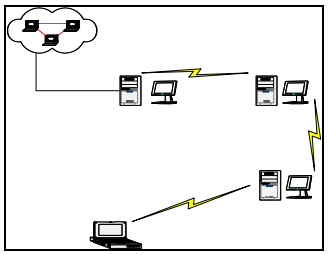
Gambar 19 memperlihatkan hasil PING 2 hop dari Adhoc-PC2 ke sebuah *node* pada LAN dengan *IP Address* 192.168.10.106 melalui Adhoc-GW. Pada percobaan 2 hop ini juga belum terdapat paket yang hilang. Dari 1176 paket yang dikirim, 1176 paket diterima, tetapi terdapat 183 paket duplikasi. Ini menunjukkan kinerja AODV cukup bagus.

```
[root@Adhoc-PC2 root]# ping 192.168.10.106
PING 192.168.10.106 (192.168.10.106) 56(84) bytes of data.
...
---dipotong---
...
64 bytes from 192.168.10.106: icmp_seq=1166 ttl=126 time=2.80 ms
64 bytes from 192.168.10.106: icmp_seq=1167 ttl=126 time=2.78 ms
64 bytes from 192.168.10.106: icmp_seq=1168 ttl=126 time=2.51 ms
64 bytes from 192.168.10.106: icmp_seq=1169 ttl=126 time=2.70 ms
64 bytes from 192.168.10.106: icmp_seq=1170 ttl=126 time=2.70 ms
64 bytes from 192.168.10.106: icmp_seq=1171 ttl=126 time=2.51 ms
64 bytes from 192.168.10.106: icmp_seq=1172 ttl=126 time=2.51 ms
64 bytes from 192.168.10.106: icmp_seq=1173 ttl=126 time=2.92
```

Gambar 19. Rata-rata *Bandwidth* Pada Percobaan 2 Hop dari Adhoc-PC2 ke *node* lain

```
[root@Adhoc-PC2 root]# traceroute Adhoc-GW
traceroute to 192.168.180.101 (192.168.180.101), 30 hops
max, 38 byte packets
1 Adhoc-PC1 (192.168.180.103) 1.762 ms 1.166 ms  1.324ms
2 Adhoc-GW  (192.168.180.101) 4.609 ms 2.492 ms  2.408ms
```

Gambar 20. Pengecekan 2 Hop



Gambar 21. Skenario Percobaan 3 Hop

Untuk membuktikan kebenaran proses hop (2 hop), digunakan perintah traceroute seperti terlihat pada Gambar 20. Perintah ini dieksekusi pada Adhoc-PC2 dengan tujuan Adhoc-GW. Terlihat bahwa dari Adhoc-PC2 untuk mencapai tujuan Adhoc-GW harus melalui 2 hop, yaitu Adhoc-PC1 kemudian Adhoc-GW.

Percobaan 3 Hop

Pada percobaan 3 hop Adhoc-NB diaktifkan dan AODV dijalankan seperti Gambar 21. Untuk menguji konektifitasnya, dilakukan PING dari Adhoc-NB ke sebuah *node* yang berada pada LAN. Pengujian lain yang dilakukan pada percobaan 3 hop ini adalah *node* Adhoc-NB menjalankan aplikasi *web browser* (Internet).

Gambar 22 memperlihatkan rata-rata *bandwidth* yang didapat oleh Adhoc-NB dengan Adhoc-GW melalui 3 hop adalah 1,24 Mbps.

Gambar 23 memperlihatkan hasil pengujian koneksi yang dilakukan oleh Adhoc-NB dengan secara terus-menerus mengirimkan paket Ping. Terlihat walaupun banyak paket data yang terduplikat, tetapi jumlah paket yang hilang masih normal. Dari 456 paket yang dikirim, 453 paket diterima dengan 85 paket duplikasi dan 3 paket hilang.

```
[  5]  0.0- 1.0 sec    216 KBytes  1.77 Mbits/sec
[  5]  1.0- 2.0 sec    200 KBytes  1.64 Mbits/sec
[  5]  2.0- 3.0 sec    136 KBytes  1.11 Mbits/sec
[  5]  3.0- 4.0 sec    168 KBytes  1.38 Mbits/sec
[  5]  4.0- 5.0 sec    136 KBytes  1.11 Mbits/sec
[  5]  5.0- 6.0 sec    112 KBytes   918 Kbits/sec
[  5]  6.0- 7.0 sec    104 KBytes   852 Kbits/sec
[  5]  7.0- 8.0 sec    168 KBytes  1.38 Mbits/sec
[  5]  8.0- 9.0 sec    136 KBytes  1.11 Mbits/sec
[  5]  9.0-10.0 sec    136 KBytes  1.11 Mbits/sec
[  5]  0.0-10.0 sec   1.48 MBytes  1.24 Mbits/sec
```

Gambar 22. Rata-rata *Bandwidth* Pada Percobaan 3 Hop

```
[root@Adhoc-NB root]# ping 192.168.180.101
PING 192.168.180.101 (192.168.180.101) 56(84) bytes of data.
64 bytes from 192.168.180.101: icmp_seq=1 ttl=64 time=2.41 ms
64 bytes from 192.168.180.101: icmp_seq=2 ttl=64 time=2.34 ms
64 bytes from 192.168.180.101: icmp_seq=3 ttl=64 time=2.32 ms
64 bytes from 192.168.180.101: icmp_seq=4 ttl=64 time=2.30 ms
64 bytes from 192.168.180.101: icmp_seq=5 ttl=64 time=2.27 ms
64 bytes from 192.168.180.101: icmp_seq=6 ttl=64 time=2.25 ms
64 bytes from 192.168.180.101: icmp_seq=7 ttl=64 time=2.23 ms
64 bytes from 192.168.180.101: icmp_seq=8 ttl=64 time=2.21 ms
64 bytes from 192.168.180.101: icmp_seq=9 ttl=64 time=2.18 ms
64 bytes from 192.168.180.101: icmp_seq=10 ttl=64 time=2.17 ms
...
...{--dipotong--}
...
ping statistics ---
456 packets transmitted, 453 received, +85 duplicates,
0% packet loss, time 459548ms
rtt min/avg/max/mdev = 4.007/11.242/1353.023/62.196 ms,
pipe 2
```

Gambar 23. *Output* PING 3 Hop

```
[root@Adhoc-NB root]# traceroute Adhoc-GW
traceroute to 192.168.180.101 (192.168.180.101), 30 hops max, 38 byte
packets
  1     Adhoc-PC2 (192.168.180.102)  4.213 ms  2.589 ms  2.956 ms
  2     Adhoc-PC1 (192.168.180.103)  6.034 ms  2.595 ms  4.001 ms
  2     Adhoc-GW  (192.168.180.101)  4.145 ms  4.712 ms  4.781 ms
```

Gambar 24. Pengecekan Percobaan 3 Hop

Seperti percobaan-percobaan sebelumnya, untuk membuktikan kebenaran terjadinya 3 hop digunakan traceroute, hasil yang didapat bisa dilihat pada Gambar 24.

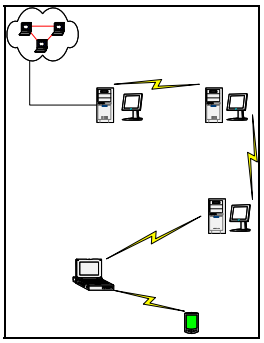
Terbukti bahwa dari *node* Adhoc-NB menuju Adhoc-GW melalui 3 hop, yaitu Adhoc-PC1, Adhoc-PC2 dan Adhoc-GW.

Percobaan 4 Hop

Percobaan 4 hop adalah percobaan terakhir yang dilakukan sesuai skenario awal. Untuk melakukan percobaan 4 hop ini, *node* Adhoc-PDA diaktifkan seperti terlihat pada Gambar 25.

Pada skenario percobaan 4 hop ini, semua *node* aktif mengirimkan paket PING ke Adhoc-GW. Dari hasil pemantauan *bandwidth* didapat rata-rata hanya 691 Kbps seperti terlihat pada Gambar 26.

Selain terjadi penurunan *bandwidth*, jumlah paket yang hilang juga makin banyak seperti terlihat pada Gambar 27. Dari 1591 paket yang dikirim, hanya 1580 diterima dengan 615 paket duplikat, berarti ada 11 paket yang hilang.



Gambar 25. Skenario Percobaan 4 Hop

```
[  3]  0.0- 1.0 sec    144 KBytes   1.18 Mbits/sec
[  3]  1.0- 2.0 sec   80.0 KBytes    655 Kbits/sec
[  3]  2.0- 3.0 sec   88.0 KBytes    721 Kbits/sec
[  3]  3.0- 4.0 sec   80.0 KBytes    655 Kbits/sec
[  3]  4.0- 5.0 sec   64.0 KBytes    524 Kbits/sec
[  3]  5.0- 6.0 sec   16.0 KBytes    131 Kbits/sec
[  3]  6.0- 7.0 sec   96.0 KBytes    786 Kbits/sec
[  3]  7.0- 8.0 sec   88.0 KBytes    721 Kbits/sec
[  3]  8.0- 9.0 sec   96.0 KBytes    786 Kbits/sec
[  3]  9.0-10.0 sec    104 KBytes    852 Kbits/sec
[  3]  0.0-10.2 sec    864 KBytes    691 Kbits/sec
```

Gambar 26. Rata-rata *Bandwidth* Pada Percobaan 4 Hop

Gambar 28 memperlihatkan bahwa dari *node* Adhoc-PDA ke Adhoc-GW melalui 4 hop, yaitu Adhoc-NB, Adhoc-PC2, Adhoc-PC1 dan Adhoc-GW.

Hasil Analisa Percobaan

Percobaan telah dilakukan di ruang Lab. Mercator lantai III Gedung Pascasarjana Fakultas Teknik Universitas Indonesia. Seperti dijelaskan sebelumnya, salah satu tujuan dari penelitian ini adalah membandingkan hasil simulasi dan *testbed*. Kinerja dari protokol routing AODV yang diukur meliputi :

- Perbandingan Paket Terkirim (*Packet Delivery Ratio*),
- *End-to-End Delay*, dan
- *Routing Overhead*.

AODV bisa berpengaruh secara langsung pada *outcome*. Ketika tidak didapatkan rute ke tujuan (*destination*), contoh disaat sebuah *node* menunggu pesan RREP, maka paket data yang akan ditransmisikan akan dimasukkan ke *buffer*. Jika ternyata sebuah rute tidak ditemukan oleh RREQ_RETRIES, maka semua paket data yang berada pada *buffer* akan dibuang (*drop*) dan pesan “Destination Unreachable” akan dikirim.

Dari Gambar 29 terlihat perbandingan paket terkirim cukup stabil pada semua percobaan pada situasi yang sama bisa didapat rata-rata 99,57% paket data terkirim.

Error bar atau *confidence interval* dari perbandingan paket terkirim dapat dihitung berdasarkan rumus [16] :

```
@Adhoc-PDA root# ping 192.168.180.101
PING 192.168.180.101 (192.168.180.101) 56(84) bytes of data.
64 bytes from 192.168.180.101: icmp_seq=1 ttl=64 time=2.41 ms
64 bytes from 192.168.180.101: icmp_seq=2 ttl=64 time=2.34 ms
...
-dipotong-
...
64 bytes from 192.168.180.101: icmp_seq=1587 ttl=61 time=81.8 ms(DUP!)
64 bytes from 192.168.180.101: icmp_seq=1587 ttl=61 time=84.8 ms(DUP!)
64 bytes from 192.168.180.101: icmp_seq=1588 ttl=61 time=7.00 ms
64 bytes from 192.168.180.101: icmp_seq=1589 ttl=61 time=7.15 ms
64 bytes from 192.168.180.101: icmp_seq=1590 ttl=61 time=8.75 ms

--- 192.168.180.101 ping statistics ---
1591 packets transmitted, 1580 received, +615 duplicates, 0% packet loss, time 1591004ms
rtt min/avg/max/mdev = 5.558/22.606/448.781/42.409 ms, pipe 2
```

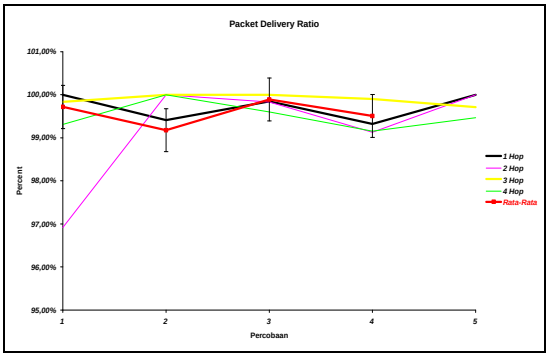
Gambar 27. *Output PING* 4 Hop

```
@Adhoc-PDA root# traceroute 192.168.180.101
traceroute to 192.168.180.101 (192.168.180.101), 30 hops max,
38 byte packets
 1  Adhoc-NB (192.168.180.107) 2.192 ms 2.311 ms 1.881 ms
 2  Adhoc-PC2(192.168.180.102) 3.830 ms 3.742 ms 3.897 ms
 3  Adhoc-PC1(192.168.180.103) 4.884 ms 4.725 ms 4.887 ms
 4  Adhoc-GW (192.168.180.101) 16.955 ms 7.533 ms 8.830 ms
```

Gambar 28. Pengecekan 4 Hop

Tabel 4. Perbandingan Ping Sukses

Jumlah Hop	Uji ke-	Paket			Persen
		Kirim	Terima	Duplikat	
1	1	390	390	1	100.00%
	2	1020	1014	1	99.41%
	3	3939	3933	113	99.85%
	4	1035	1028	6	99.32%
	5	1096	1096	22	100.00%
2	1	1589	1540	23	96.92%
	2	1049	1049	128	100.00%
	3	1188	1186	150	99.83%
	4	1033	1024	97	99.13%
	5	1176	1176	183	100.00%
3	1	1825	1822	230	99.84%
	2	1070	1070	192	100.00%
	3	1016	1016	122	100.00%
	4	1023	1022	94	99.90%
	5	1051	1048	174	99.71%
4	1	1591	1580	615	99.31%
	2	827	827	153	100.00%
	3	1257	1252	149	99.60%
	4	1185	1175	586	99.16%
	5	1129	1123	429	99.47%



Gambar 29. Perbandingan Paket Terkirim pada *Testbed*

$$s = \sqrt{\frac{n\sum_{t=1}^{n} x_t^2 - (\sum_{t=1}^{n} x_t)^2}{n(n-1)}} \quad \text{(Pers. 1)}$$

$$\overline{X} \pm t_{\alpha/2} \frac{s}{\sqrt{n^t}} \quad \text{(Pers. 2)}$$

Dengan :

$\overline{X}$ = Rata-rata

t = Koefisien interval (1.96 untuk *confidence interval* 95%)

s = Standar deviasi dari n percobaan

n = Jumlah percobaan

Sehingga didapat hasil seperti terlihat pada Tabel 5.

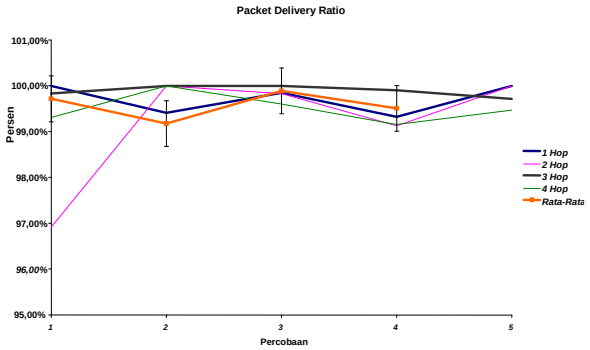
End-to-end delay adalah jumlah waktu yang digunakan oleh sebuah paket ketika dikirim oleh sebuah *node* dan diterima di *node* tujuan. *End-to-end delay* merupakan jumlah dari waktu pengiriman, propagasi, proses dan antrian dari suatu paket pada setiap *node* di jaringan.

Faktor pertama yang mempengaruhi *end-to-end delay* adalah waktu untuk menemukan rute. Hal ini berguna sebelum pesan yang akan dikirim, *node* (*source*) harus tahu terlebih dahulu jalur atau rute untuk mencapai tujuan (*destination*). Faktor lain yang mempengaruhi *end-to-end delay* adalah *delay* proses. Ketika *node* yang ditengah menerima sebuah pesan, *node* tersebut akan menganalisa *header* untuk mengetahui untuk siapa paket ditujukan, dan kemudian mengecek *node* untuk menentukan kemana harus meneruskan paket tersebut.

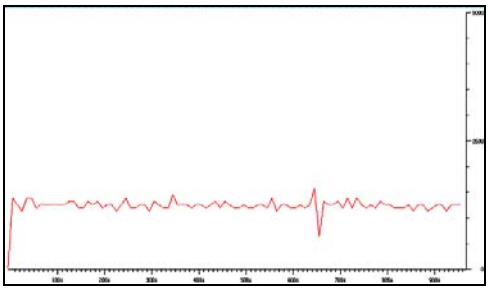
Tabel 5. Rata-Rata *End-to-End Delay* pada *Testbed*

Jumlah Hop	Uji ke-	Jumlah Waktu*	Jumlah Paket	End-to-end Delay*
1	1	392.893	390	1.007.42
	2	1.029.164	1.020	1.008.98
	3	3.977.352	3.939	1.009.74
	4	1.044.274	1.035	1.008.96
	5	1.105.950	1.096	1.009.08
2	1	1.603.383	1.589	1.009.05
	2	1.048.827	1.049	999.84
	3	1.187.912	1.188	999.93
	4	1.032.656	1.033	999.67
	5	1.176.000	1.176	1.000.00
3	1	1.825.268	1.825	1.000.15
	2	1.079.698	1.070	1.009.06
	3	1.025.121	1.016	1.008.98
	4	1.032.198	1.023	1.008.99
	5	1.060.480	1.051	1.009.02
4	1	1.591.004	1.591	1.000.00
	2	826.620	827	999.54
	3	1.256.155	1.257	999.33
	4	1.185.078	1.185	1.000.07
	5	1.128.255	1.129	999.34

* Dalam millidetik



Gambar 30. *End-to-End Delay* pada *Testbed*



Gambar 31. *Routing Overhead* pada *Testbed*

Tabel 6. *Confidence Interval* Perbandingan paket Terkirim pada *Testbed*

Jumlah Hop	Batas	
	Bawah	Atas
1	99,43%	100%
2	98,03%	100%
3	99,78%	99,99%
4	99,05%	99,72%

Tabel 6 memperlihatkan rata-rata *end-to-end delay* pada setiap percobaan *multihop*. Gambar 30. memperlihatkan hasil rata-rata *end-to-end delay* pada kelima percobaan masing-masing hop. Didapat hasil rata-rata *end-to-end delay* sebesar 1.004, 36 millidetik atau dibulatkan 1 detik.

Routing overhead adalah jumlah seluruh routing paket yang dikirimkan selama percobaan berlangsung, baik bagi paket-paket yang ditransmisikan melalui 1 hop atau *multiple* hop. Terlihat pada Gambar 31 grafik *routing overhead* protokol *routing* AODV-UU cukup fluktuatif berkisar antara 1.000 ~ 1.500 Byte per detik.

Testbed telah dilakukan di dalam ruangan dengan pertimbangan faktor lingkungan yang bisa berefek terhadap kinerja AODV. Faktor-faktor itu antara lain suhu ruangan, halangan, frekuensi radio, dan suara

bising. Hasil yang didapat adalah perbandingan paket terkirim (*packet delivery ratio*) 99,57%, *end-to-end delay* 1,004 detik dan *routing overhead* 1.360,36 Byte.

Perbedaan hasil pengukuran pada simulasi dan *testbed* dapat disebabkan pada simulasi menghilangkan beberapa parameter, seperti *delay* pada masing perangkat keras yang diasumsikan nol, sedang pada kenyataannya (*testbed*) setiap perangkat keras memiliki *delay*.

Penelitian kedepan yang dapat dilakukan antara lain: *Cross compilation* AODV-UU untuk PDA iPAQ h5550 masih menjadi masalah pada Linux Familiar dengan kernel 2.4.19, keamanan pada jaringan *ad hoc hybrid*, dan implementasi *Mobile* IPv6 pada MANET.

4. Kesimpulan

Simulasi yang telah dilakukan menghasilkan rata-rata perbandingan paket terkirim (*packet delivery ratio*) 99,89%, *end-to-end delay* 0.14 detik dan *routing overhead* 1.756,61 Byte.

Daftar Acuan

[1] C. Kopp, “Ad hoc Networking”, Background Article, Published in ‘System’, (2002) p.33-40.
[2] Hamidian, Master’s Thesis, Departement of Communication System, Lund University, Sweden, 2003.
[3] Project MobileMAN, http://keskus.hut.fi/tutkimus/MobileMan/, 2005.
[4] J.C. Requena, J. Gutierrez, R. Kantola, J. Creado, N. Beijar, 11th International Conference of Telecommunications Brazil, 2004.
[5] C. Perkin, E.M. Belding-Royer, S. Das, Ad hoc On Demand Distance Vector (AODV) Routing, IETF Internet Draft, 2002.
[6] T. Staub, Performance Comparison of MANET Routing Protocols in Ad-hoc and Hybrid Network, Computer Science Project, University of Berne, Switzerland, 2004.
[7] Anon., http://www.isi.edu/nsnam/ns, 2005.
[8] A.B. Wirawan, E. Indarto, Mudah Membangun Simulasi dengan Network Simulator-2, Andi offset, 2004.
[9] http://www.gnuplot.info, 2005.
[10] V. Noumov, G. Thomas, Simulation of Large Ad Hoc Networks, ETH Zurich, 2003.
[11] http://www.docs.uu.se/scanet/aodv/, 2005.
[12] http://www.handhelds.org, 2004.
[13] http://www.openxtra.co.uk/support/howto/ethereal-display-filter.php, 2005.
[14] http://dast.nlanr.net/Projects/Iperf/, 2005.
[15] http://www.linuxguruz.com/iptables/howto/, 2005.
[16] R. Jain, The Art of Computer System Performance Evaluation Performance Analysis, John Willey, 1991.