AODV-UI Proof of Concept on MIPS-based Wireless Router

Boma Anantasatya Adhi, Ruki Harwahyu, Abdusy Syarif, Harris Simaremare, Riri Fitri Sari, and Pascal Lorenz

Original scientific paper

Abstract: **AODV routing protocol facilitates changing and simple-to-setup network environment. It helps setting up a network without sufficient infrastructure, such as in disaster area. Development of AODV protocol has gathered a worldwide research interest. However, not many researches implement AODV routing protocol in real mobile nodes and real MANET. In addition, real implementation deals with other works concerning underlying protocol, firmware and hardware configuration, as well as detailed topology both in logical and physical arrangement. This work aims to implements Ad-hoc On-demand Distant Vector – particularly University of Indonesia AODV (AODV-UI) routing protocol on low-end inexpensive generic wireless routers as a proof of concept. AODV-UI is an improved version of AODV routing protocol that implements gateway interconnection and reverse route capability. This routing protocol has been previously successfully tested in NS-2. In this work, current AODV-UI protocol is ported to OpenWRT + MIPS (Microprocessor without Interlocked Pipeline Stages) little endian architecture then tested on the real networking environment. Underlying media access layer is also altered to provide the protocol greater control over the network. Performance of this implementation is measured in terms of energy consumption, routing overhead, end-to-end delay, protocol reliability and packet delivery ratio.**

***Index terms*: AODV-UI, MIPS, OpenWRT, embedded wireless router**

I. INTRODUCTION

AODV routing protocol facilitates changing and simple-to-setup network environment [1]. It can create multi-hop routing between participating mobile nodes forming continuous ad-hoc network with a little user interference. This is the principle of mobile ad-hoc network (MANET). Mobile Ad-hoc Network (MANET) is a non-infrastructure network that consists of a collection of nodes that can communicate each other independently [2]. These advantages of MANET will help setting up a network without adequate infrastructure, such as in disaster or catastrophe area, battle field, and hazard area.

In MANET, unpredictable nodes mobility yields dynamic topology that requires a certain mechanism to keep the connection. Furthermore, there is no centralized access point as in traditional network [3,4]. Because of the absence of administrative nodes to control the network, every mobile node exist in the network is responsible to ensure a reliable network operation.

Development of AODV protocol has gathered a worldwide research interest due to the fact the AODV protocol can run well in high mobility and high traffic communication. A new variant of AODV protocol called AODV-UI was introduced in 2011. This newly introduced protocol implements reverse route method and gateway mode to improve AODV’s performance as well as improvement in energy consumption.

However, as this Ad-hoc routing protocol obtains more interests, only a small number of research implements AODV routing protocol in real world use real mobile nodes and real MANET. [5] implements Better Approach To Mobile Adhoc Networking (B.A.T.M.A.N) in Android devices. However, this implementation is conducted to test the web caching mechanism without detailed assessment toward the underlying routing protocol. [6] presents the implementation of new approach for consistent and efficient name resolution using adaptive routing techniques in Linux-based platform using the Click Modular Router and its validation using the Network Simulator 3. The study lacks of detailed reports about the routing performance. In [7], an exploration of Atheros ath5k driver for wireless ad-hoc routers is shown. Although this study does not specify any routing protocol, it elaborates the technical detail in MAC layer, its source code for the driver, and its realistic scenarios.

None of the above studies are specifically assess AODV protocol and its implementation in real device. Real world implementation is important since simulation environment such as NS-2 always gives underlying configuration provided flawlessly. By conducting real world implementation, we deal with other works regarding underlying protocol, firmware and hardware configuration, as well as detailed topology both in logical and physical adjustment. We can also observe the real performance further.

In this paper we present the result of analysis towards AODV-UI implementation on real devices and play their role on a real MANET. The paper elaborates the general issues that arise about MANET and AODV. Our most valuable work is porting AODV-UI into Linux based embedded system (OpenWRT) with MIPS based processor. MIPS was originally acronym for Microprocessor without Interlocked Pipeline Stages. Today, the term MIPS is generally referes

Manuscrip received on April 14, revised May 2014.

This work is part of International Joint Collaboration and Scientific Publication project between Universitas Indonesia, Indonesia and Universite Haute-Alsace, France, 2011-2013.

Boma Anantasatya Adhi, Ruki Harwahyu and Riri Fitri Sari are with the Universitas Indonesia, Indonesia. E-mails: boma@ui.ac.id, ruki.h@ui.ac.id, riri@ui.ac.id.

Abdusy Syarif, Harris Simaremare, and Pascal Lorenz are with the Universite Haute-Alsace, France. E-mails: abdusy.syarif@uha.fr, harris.simaremare@uha.fr, pascal.lorenz@uha.fr.

to a reduced instruction set computer developed by MIPS Technologies, Inc. The important implementation steps are described as well as the setup of the testing environment. The result is discussed based on the significant aspects of its performance for MANET. We compare the real-world implementation result against the previous simulation result that has been conducted by Sari et al [8] and try to discuss some important aspects for MANET.

II. LITERATURE STUDY

A. AODV

Ad-hoc On-demand Distance Vector (AODV) is a routing protocol intended to be used on mobile network. It provides a method for mobile terminals to traverse messages through their neighbors to other terminals that they cannot communicate directly.

Basically, each mobile terminal can only communicate with the terminals next to it. AODV routes the messages along by discovering the neighbors within their range and forming the routes to be used to forward the messages. The route will include other terminals outside one's range that can pass the messages to the destination. AODV chooses the shortest loop-free path. It creates new route for different destination when needed and does not require the preservation of routes that are not in active communication [9]. Furthermore, it can recover the route to a certain destination if the topology is changing due to terminal's movement or if there is an error. This is critical for mobile network.

AODV uses Bellman-Ford algorithm to computes shortest and loop-free path. It also uses sequencing method for each route entry. The destination sequence number is created by the destination to be included along with any route information it sends to requesting nodes. Routes with the greatest sequence number will be used. AODV employs several type of messages encapsulated in UDP and normal IP packet. According to the RFC35611 [1], these messages are Route Request (RREQs), Route Replies (RREPs), Route Errors (RERRs), and RREP-ACK.

The format of RREQ message is illustrated in Table I. In this message, field Type contains value 00000001 (1). Field J contains Joint flag that is reserved for multicast. Field R contains Repair flag. Field G contains Gratuitous RREP flag, indicating whether a gratuitous RREP should be unicast to the terminal specified in the Destination IP Address field. Field D contains Destination-only flag, indicating only the destination may respond to this RREQ. Field U for Unknown sequence number indicates that the destination sequence number is unknown. Field reserved contains zeros, as it will be ignored. Field RREQ ID contains a sequence number uniquely identifying the particular RREQ for each originating terminal.

TABLE I
RREQ MESSAGE FORMAT

Byte 1	Byte 2	Byte 3	Byte 4
Type	Flags + Reserved		Hop Count
RREQ ID			
Destination IP Address			
Destination Sequence Number			
Originator IP Address			
Originator Sequence Number			

TABLE II
RREP MESSAGE FORMAT

Byte 1	Byte 2	Byte 3	Byte 4
Type	Flags + Reserved + Prefix Size		Hop Count
Destination IP Address			
Destination Sequence Number			
Originator IP Address			
Lifetime			

The format of RREP message is illustrated in Table II. In this message, field Type contains value 00000010 (2). This message has R field indicating Reply flag that is used for multicast, A field indicating whether Acknowledgment is required, Prefix Size specifies that the indicated next hop may be used for any nodes with the same routing prefix as the requested destination, Lifetime contains time in milliseconds for which receiving terminal consider the route to be valid.

The format of RERR message is illustrated in Table III. In this message, field Type contains value 00000011 (3). This message lists unreachable IP address and uses field N indicating No delete flag. This flag is set when a node has performed a local repair of a link, and upstream nodes should not delete the route.

The last message is RREP-ACK. The format of RREP-ACK message is illustrated in Table IV. In this message, field Type contains value 00000100 (4). This message is used to reply a RREP message that has its field A set to 1.

TABLE III
RERR MESSAGE FORMAT

Byte 1	Byte 2	Byte 3	Byte 4
Type	Flags + Reserved		Dest. Hop Count
Unreachable Destination IP Address			
Unreachable Destination Sequence Number			
... *<more unreachable destination & sequence>* ...			

TABLE IV
RREP-ACK MESSAGE FORMAT

Byte 1	Byte 2
Type	Reserved

A route is required when an end terminal want to send messages. AODV does not perform anything as long as both ends of the connection have valid rote to each other. If there is no route available on the routing table, this terminal broadcasts a RREQ to find a route to a certain destination. A route can be determined when the RREQ reaches either the destination itself or an intermediate node with a valid route to the destination. Valid means the route entry for the destination whose associated sequence number is equal or greater than the sequence number contained in the RREQ. When a route is found, RREP is sent back to the sender of RREQ. Each node receiving the RREQ caches a route back to the sender of the RREQ, so that the RREP can be sent back as unicast message.

Each terminal monitors the next hops for its link status in active routes entry. When a link in an active route entry is broken, the terminal who detects it sends RERR message to notify other nodes that a route is no longer available. The RERR message indicates those destinations are no longer reachable because of the broken link. Each terminal keeps a precursor list, containing the IP address of neighbors who use this terminal as a next hop to forward the message. The RERR message is forwarded to the terminal listed in precursor list, which is acquired during the generation of RREP message.

AODV builds routing table on-demand and does not keep all routes for long time. However, AODV do keeps route table information temporarily for reverse paths towards nodes originating RREQs. AODV uses the following fields with each route table entry [1]:

- Destination IP Address
- Destination Sequence Number
- Valid Destination Sequence Number flag
- Other state and routing flags (e.g., valid, invalid, repairable, being repaired)
- Network Interface
- Hop Count (number of hops needed to reach destination)
- Next Hop
- List of Precursors
- Lifetime (expiration or deletion time of the route)

The unusual thing in this routing protocol is the sequence number. It is a number maintained by each originating terminal that always increment. This number is used to determine the validity of the information contained from the originating terminal. Managing the sequence number is crucial to avoid routing loops. This value will go back to zero after reaching its maximum value.

Every destination in route table includes the latest unique sequence number. This number is updated once the terminal receives RREQ, RREP, or RERR that contains information about this destination. While maintaining the route for this destination, each terminal will increment its own sequence number before this terminal sends a RREQ. Sequence number also changed to the maximum of its current sequence number and the destination sequence number in the RREQ packet after this node receives a RREQ, immediately before sending a RREP.

When the link to a next hop is lost or expired, the terminal will check every route that uses this next hop. The terminal increments the sequence number and marks the route as invalid. This mark can only be changed when the link connected again or by receiving other valid AODV message.

B. AODV Implementation Comparison

As the fundamental studies, we include some AODV implementations that are available for public. They are AODV-UU, AODV-UCSB, Kernel AODV, and AODV-UIUC.

AODV-UU is an AODV implementation that is developed in Upsala University. This AODV implementation is developed to be used in Linux user-space and also have been ported on NS2 simulator environment. AODV-UU uses kernel module to communicate with Netfilter and detect the event. All routing protocol processes and logics are processed by a daemon that resides in user-space. AODV-UU improves the performance of Hello packets to support unidirectional link support and signal quality threshold for received packets.

AODV-UCSB is an AODV implementation that is developed at the University of California in Santa Barbara. It was formerly developed by directly modifying the kernel without requiring kernel module or any user-space applications. Later, this protocol uses kernel module from AODV-UU release version 0.4. On its latest release, most of AODV-UCSB logic is implemented in user-space daemon. The improvement that has been made includes options for the number of Hello message to connect to neighbor device. It avoids device in creating routes to neighbors based on a single spurious message reception.

AODV-UIUC is another user-space implementation of AODV that uses Netfilter and the additional wrapper called Ad-hoc Support Library (ASL). ASL is a user-space library which provides an API to facilitate implementation of routing protocols for wireless ad-hoc networks in Linux. Ad-hoc Support Library provides a simple API to implement on-demand or reactive ad-hoc routing protocols in Linux. It is implemented as a userspace library. No kernel modifications are required. AODV-UIUC is developed in University of Illinois at Urbana-Champaign. This implementation completely separates routing function and forwarding function. Routing functionality is handled in the user-space daemon meanwhile packet forwarding tasks is done by the kernel. By this method, packet forwarding can be processed immediately and the packet that should traverse between kernel-space and user-space is fewer.

For these three implementations, most of the protocol logics are done in user-space. All interesting packets in the kernel space are captured using Netfilter then forwarded to userspace. Netfilter is a framework that enables packet filtering, network address and other packet mangling. Netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.

Kernel AODV is developed at the National Institute of Standards and Technology (NIST). Kernel-AODV uses Netfilter, whereas all of the routing protocol logic is placed inside the kernel module instead of user-space daemon. This placement improves the performance of the implementation, in terms of packet handling, since no packets are required to traverse from the kernel to the user-space. This implementation also supports Internet gateway, multiple interfaces and a basic multicast protocol. By using this kernel-level implementation, very low latency is expected.

C. MIPS Architecture

MIPS (formerly accronym for Microprocessor without Interlocked Pipeline Stages) is a 32 and 64 bit Instruction Set Architecture (ISA) developed by MIPS Technology (MIPS technology has been acquired by Imagination since 2013) [10]. MIPS ISA commonly found in embedded system environment, especially embedded network devices such as routers, residential gateway, and wireless access point[11]. As explained before, in this experiment AODV-UI is ported to MIPS ISA due to the fact that many wireless capable embedded network devices suitable for MANET environment are powered by this architecture.

Extensive development environment for this architecture can be easily obtained both as commercial or open source solution. This architecture is also supported in the mainline Linux kernel. The GNU C Compiler (gcc) target, tool chain and binary utilities for this architecture is also exist in mature state[11]. Several flavor of linux distribution for embedded network device such as OpenWRT, DD-WRT, Tomato and FreeWRT has also been ported to this architecture. This reason also support our decision to port AODV-UI to this architecture.

D. OpenWRT

OpenWRT is a highly extensible GNU/Linux distribution for embedded devices [12]. It is a free operating system designed for wireless router. This operating system is developed based on Linux and uses Linux kernel. It has wide compatibility with many embedded CPU architecture and has wide support for many wireless chipset as it is designed intentionally to handle wireless router's tasks. By utilizing this operating system, developer can freely customize the wireless router's capability.

OpenWRT has its own writable file system and package management system. It contains hundreds software packages and provides free access to the source code to include more add-on packages. The OpenWRT source code is oriented towards developers by providing a development kit called OpenWRT Software Development Kit (OpenWRT SDK), which enabled us to develop adaptation part for the AODV-UI source codes.

III. PROPOSED IMPLEMENTATION

A. AODV UI

One of the disadvantages of AODV protocol is that the source node must re-initiate communications by running route discovery procedure, and try to find new path communication, when communication between nodes is lost due to the high mobility of nodes. AODV-UI [13] was developed to overcome this problem.

AODV-UI is a new variant of AODV routing protocol that is proposed by Sari et al [8]. This proposed protocol combines gateway mode and reverse route. The main feature of AODV-UI is it can be used in hybrid ad-hoc networks and it has some improvement in routing overhead and energy consumption. This protocol has an algorithm to determine which nodes as an intermediate node. This protocol also includes energy as a parameter in selecting node. At the latest development, it includes a method for malicious node detection and removal to ensure secure path in public MANET [8].

The gateway mode enables a node to be set as a gateway. In real implementation, this node can be a stationary device backed with wired connection as the gateway. Other mobile node on the MANET will use this gateway to connect to the outside network such as public internet. This gateway mode is adopted from AODV+ [14]. AODV+ has been designed to achieve routing communication between a node in an ad-hoc network and a node in a wired network or infrastructure. This variant of AODV uses hybrid gateway discovery mode since it combines two gateway discovery models, i.e. reactive and proactive. The reactive gateway discovery is initiated by ad-hoc node to create or update a routing table to a gateway. In this discovery mode, mobile node will broadcast a route request (RREQ) message with "I" flag (RREQ_I). It will be processed only by the gateways addressed by this message. In the other hand, the proactive gateway discovery is initiated by the gateway itself. This gateway can be a mobile node or other static terminal with wireless connection. It broadcasts a gateway advertisement (GWADV) message periodically to ad-hoc network, so a mobile node that receives the GWADV will update its route entry.

The reverse route is adopted from R-AODV [16,17]. R-AODV provides solutions for the MANET topology that is changing rapidly, since frequent change of network topology is a tough challenge for many important issues, such as routing protocol robustness and resilience performance degradation. This variant of AODV provides reverse route by trying multiple route replies. This protocol has been proved in reducing path fail correction message.

In R-AODV, RREQ message has additional 4-bytes field for timestamp. For the route discovery, immediately after receiving the first RREQ message, the node broadcast reverse request (R-RREQ) message rather than sending unicast RREP. This is the main differences of this variant against the standard AODV. The format of R-RREP message is illustrated in Table V.

TABLE V
RREP MESSAGE FORMAT IN R-AODV

Byte 1	Byte 2	Byte 3	Byte 4
Type	Reserved		Hop Count
R-RREP ID			
Destination IP Address			
Destination Sequence Number			
Originator IP Address			
Reply time			

The processing procedure of this message is same with the RREQ message in AODV. However, originating nodes will receive multiple R-RREQ suggesting a valid path toward the destination. Originating nodes will choose the best path to forwarding the packet.

By broadcasting R-RREQ, this protocol has more control packet overhead. Nevertheless, this overhead is smaller compared to the standard AODV that uses only single reply message. For instance, in ad-hoc network has N number of terminals with M nodes participate to discover a routing path, required number of control messages to discover routing path for AODV (P) if it does not fail in first try is expressed in equation (1)

$$P = M - 1 + t \qquad (1)$$

where t is the number of nodes relied on route reply message.

If source node fails in the first try because route reply message could not arrive, the node will re-initiates path discovery. It means the number of control messages increase by the number of attempts, as expressed in equation (2).

$$\mathrm{P} = \mathrm{c}(\mathrm{M} - 1 + \mathrm{t}) \qquad (2)$$

where c is the number of attempt for route discovery.

If we assume that R-AODV has at least one stable path by a RREQ, then the number of control messages for R-AODV (Q) is expressed in equation (3).

$$\mathrm{Q} = 2\mathrm{M} - 2 \qquad (3)$$

We can conclude when c>1, standard-AODV causes more packet overhead than the case of c=1 on R-AODV routing. This condition is likely to be experienced as mentioned in [18], when the number of nodes is 100 and the number of flows is 50, 14% of total RREP messages are lost in standard AODV.

B. Hardware and Software Stack

As previously mentioned, AODV-UI routing protocol is intended to be implemented on low-end and inexpensive wireless routers. This is done by replacing the original firmware of the wireless router with OpenWRT and adding AODV-UI capability on it. We choose OpenWRT over other firmware codebase such as DD-WRT and FreeWRT because it offers greatest stability, general Linux utilities and modular compiling system that makes it easier for development.

OpenWRT Backfire 10.03 is used for TP-Link MR3220 v1.0 wireless router as the first platform, and OpenWRT White Russian RC4 is used for Linksys WRT54GL v1.1 as the second platform. TP-Link MR3220 uses AP99 Atheros AR7241 400MHz MIPS Processor and Linksys WRT54GL uses Broadcom 5352 200MHz MIPS Processor.

AODV-UI is built based on AODV-UU codebase which is then cross compiled to MIPS architecture. AODV-UU has provided a working AODV routing protocol and complete gateway mode. Reverse route is implemented by changing unicast RREP message on AODV-UU to broadcast R-RREQ message according to AODV-UI original specification. R-RREQ message format used in this implementation slightly differs from original AODV-UI's R-RREQ specification. This implementation uses broadcasted AODV-UU's RREP format instead. Therefore no change needed on any node to process R-RREQ message as it is identical to RREP.

Since R-RREQ is broadcasted, there is chance that a node receives a lot of R-RREQ for a single RREQ which it has sent previously. Therefore only first R-RREQ messages on originator will be processed. This decision is based on assumption that the fastest R-RREQ message path is the most efficient path to reach the destination.

Fig. 2 depicts current implementation stacks of AODV-UI. Most of the logic and routing algorithm codes run on userspace to minimize effect of defective code on system stability. Kernel space implementation is limited to routing table manipulation codes only.

Since the current implementation is based on AODV-UU codebase, it is available on several generic Linux platforms: such as x86, AMD64, MIPS little and big endian, and ARM.

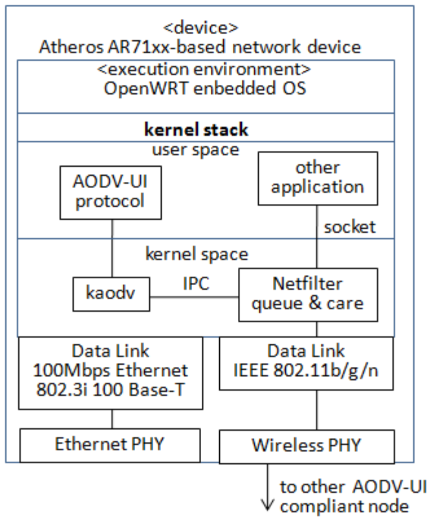

Fig. 1 Deployment diagram

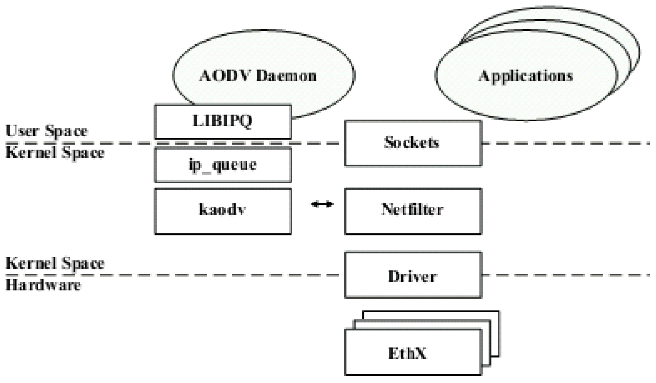

Fig. 2 Netfilter Architecture used in the implementation

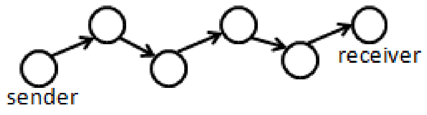

Fig. 3 Node topology 1

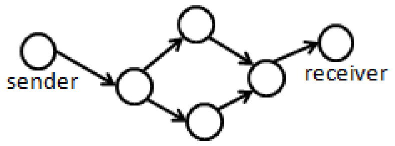

Fig. 4 Node topology 2

Current kernel module utilize Netfilter architecture stack to manipulate the network packets in kernel space as shown in Fig. 1, therefore kernel module compatibility list is limited to kernel 2.6.32 and older. Implementation on newer kernel without Netfilter is in progress.

Compilation is done on Debian Squezy with OpenWRT SDK, gcc4.4.5 linaro, Debian's build essential and full development environment installed, including binutils, glibc, m4, bison, autoconf, and automake.

C. Experiment Setup

In this experiment, AODV-UI is tested on real wireless networking environment. All tests were done indoor with no other wireless infrastructure operating at adjacent channel nearby. All nodes operated in 802.11g mode at channel 11. Wireless transmit power of all nodes were reduced to 25% due to space restriction. Smaller 2dbi antennas were also used to further reduces the effective range of each node so that each node can only communicate with other nodes (less than 1% packet loss) in < 12 m radius.

Default configuration in this experiment is follows:

- Inter-node distance is 10m
- Number of node is 5 (4 units of platform 2 and a laptop based reference platform)
- No background traffic

This default configuration is used in all tests unless stated otherwise. There were several test scenarios in this experiment:

- Basic networking test: this test utilized 2 units of platform 2 nodes directly connected via MANET.
- Implementation comparison test utilize 2 units of platform 2 nodes and an AODV-UI enabled laptop.
- Basic AODV functionality: this test utilized 4 units of platform 2 nodes and an AODV-UI enabled laptop. All nodes were arranged as depicted in Fig. 3. The node is arranged so that each of them could only communicate with 2 closest neighbors except for the end node.

Advanced AODV-UI feature test is conducted by either utilizing topology 1 as shown in Fig. 3 or topology 2 shown in Fig. 4. Inter-node distance is varied.

IV. Result and Discussion

As the preliminary stage, we assess the MANET using simple tests to observe the basic performance of each device running AODV-UI routing protocol. We use three scenarios as follows.

A. Basic Networking Test

This test uses peer-to-peer connection intended to assess whether the ad-hoc connection runs well on the normal condition. In this test, only two nodes are used. The first node acts as traffic source and the second node acts as neighbor with an active IP address to be ping from the first node. Both nodes are placed within the range of each other. A loopback interface is activated on the second node. The first node ping this loopback interface continuously and we analyze the ping result captured using wireless sniffer device and the ping result that is directly received by the first node. This test is repeated 5 times and validates it statistically.

For this static condition, result shows that by sending 61 ping packet, average percentage of packet loss is 1.6% in platform 1 and 3.2% in platform 2.The number of out of order reply received by the first node is 0 % on both platforms. The complete assessed criteria are shown in Table VI. This result may be varying for different number of active node, different distance between communicating node and different movement pattern of the nodes.

TABLE VI
RESULT FROM PEER-TO-PEER CONNECTIVITY TEST

Parameter	Platform 1	Platform 2
Average packet sent	61	61
Average packet received	60	59
Average unordered reply	0	0
Average roundtrip time	7,2 ms	7,4 ms
Maximum roundtrip time	64,2 ms	80,2 ms
Minimum roundtrip time	1,4 ms	2.4 ms

TABLE VII
AVERAGE CONTROL PACKET FORWARDING DELAY

Destination	AODV-UU	Kernel AODV	AODV-UI
Reference platform	150 us	11 us	152 us
Platform 1	1087 us	N/A	998 us
Platform 2	445 us	N/A	483 us

TABLE VIII
ROUTE TABLE IN NODE 192.168.1.10

Destination	Gateway	Mask	Flag
192.168.1.7	0.0.0.0	255.255.255.255	UH
192.168.1.9	**192.168.1.7**	**255.255.255.255**	**UHG**
192.168.0.0	0.0.0.0	255.255.255.0	U
192.168.0.0	0.0.0.0	255.255.0.0	U

This result presents slight difference between the roundtrip delays between the two platforms. A large value is observed on maximum roundtrip time. It is likely caused by processing overhead required by AODV. This overhead requires several amount of time that is also experienced during the initial condition when the first node tried to ping the second node after all nodes reset.

B. Implementation Comparison Test

This assessment intends to compare this implementation in all used platforms and an additional general computer based system as a reference. Control packet forwarding delay is compared. This delay is measured for a RREQ packet with no entry in the routing table starting from the time it enters the device and is forwarded to next node. This test reveals the time needed by the netfilter kernel module to grab the packet content, send it to the userspace daemon, then the daemon has to seek for its destination in routing table and if not exist resend it as a new RREQ or if exist reply with RREP.

This test is considered important since this protocol is intended to be used on mobile device, which might be very limited in term of processing and power durability, yet expecting high data throughput for various communications. No routing overhead taken into account in this test.

The more powerful mobile computer system we use has AMD E-350 dual core x86_64 processor, 6GBs RAM, and 320GBs storage. It runs Linux kernel 2.4.30 and 2.6.32. It uses gcc 4.6 and gnu toolchain for helping the development of the routing protocol. The result, as shown in Table VII, denotes that Kernel AODV consume less time since packet only cross kernel-user space boundary once.

Aside from the control packet forwarding delay, we also try to observe CPU usage on both MIPS architecture and PC based x86 processor. Unfortunately we could not reliably measure the performance of the implementation in the MIPS architecture due to limited computing performance. As an example, during route formation PC based implementation has only a little more than 1% CPU usage and almost negligible. On the other hand, the less powerful MIPS based wireless router has about 30% CPU load all the time. This is largely due to extreme performance difference between the two architecture.

C. Basic AODV Functionality Tests

This assessment intends to observe whether the device can discover its neighbor along with its performance. We use two nodes and conduct the analysis in one of them. We use synchronized time server and adjusted them before the AODV process begins.

The process is to let the AODV populate its routing table with local entry (4 neighbors). We measure the time between the routing daemon is started until all four neighbor is detected. We repeat this experiment ten times in different network condition varied by its level of background traffic.

In this experiment the background traffic is generated by a custom program running on each node. The program tries to send a dummy packet to any neighboring nodes at specified rate. At 0%, no traffic is generated at all. At 100%, the program will try to generate as much packet as the wireless link can handle (pre computed before). And at 120% the program generates 20% more traffic than the wireless link capacity thus resulting in a lot of dropped packet as the buffers overflow.

The result is shown in Fig. 5. Neighbor discovery time gets higher as background traffic grows and saturated at around 55 second.

The next AODV functionality to be tested is its route formation. This test intended to observe whether AODV nodes' cooperation works well in delivering the data from the traffic source to the destination. In this test, four nodes are used, for instance we call them node A with IP address 192.168.1.7, node B with IP address 192.168.1.8, node C with IP address 192.168.1.9 and node D with IP address 192.168.1.10.

Gateway node is noted with G flag for requested node that is not directly connected. Route formation can only ignited by TCP/UDP packet without valid route in current route table. In our case, node 192.168.1.10 initially only connected directly to its neighbor, node 192.168.1.7. After node 192.168.1.10 requested a route to 192.168.1.9, AODV working together finding the path and reported it in every concerned node. In this case, route table of node 192.168.1.10 is added with an entry of 192.168.1.9. And node 192.168.1.9 is added into the routing table along with a "G" flag. Route table of node 192.168.1.10 is shown in Table VIII with a newly added entry shown in bold text, node 192.168.1.9 as a gateway. This routing table indicates that the AODV is successfully running its multi-hop route formation in ad-hoc network.

Next, packet routing delay is observed under several conditions. In this experiment, packet routing delay is time delay measured starting from a packet with destination outside the local routing table until it is sent to the next hop. This test was done using topology 1.

In Fig. 6, routing delay goes up linearly as number of hops increased. This is due the fact that each node requires time to process and forward each route request and reply. AODV-UU and AODV-UI performance is comparable.

In a real wireless networking environment, it is impossible to directly adjust packet loss efficiently. To achieve a desired packet loss ratio, the distances between nodes are adjusted to critical point where every node is barely able to communicate with each other and then adjusted accordingly. This method is note very accurate, but it is still effective to simulate bad link conditions coarsely.

Fig. 7 shows that less than 10m inter-node distance with 4 hops, there is visible correlation between routing delay and inter-node distance for both protocols. Above 10m inter-node distance as packet loss increases, each protocol starts taking more time to create a valid route. At 13m AODV-UI takes 48.23 second to route and AODV-UU timed out before making any valid route. Please mind the logarithmic vertical axis.

D. Advanced AODV-UI Feature Test

This part of the experiment focuses on features specific to AODV-UI, gateway mode and reverse route. Since AODV-UI is built based on AODV-UU, gateway mode implementation is already proven prior to this implementation. Therefore this section emphasis on testing the newly built feature: Reverse Route capability.

AODV-UI replaces unicast RREP with broadcasted R-RREQ. Broadcasted R-RREQ ensures more route replies sent to the originator. To check this capability, node topology 2 with multiple paths is used. R-RREQ shows its advantage in bad link conditions; therefore the next test is done in the critical inter-node distance, 10m-13m.

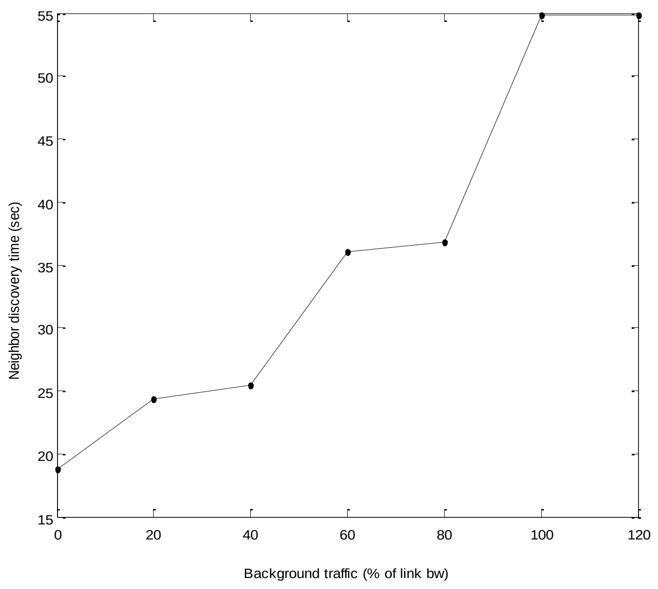

Fig. 5 Neighbor discovery time with background traffic

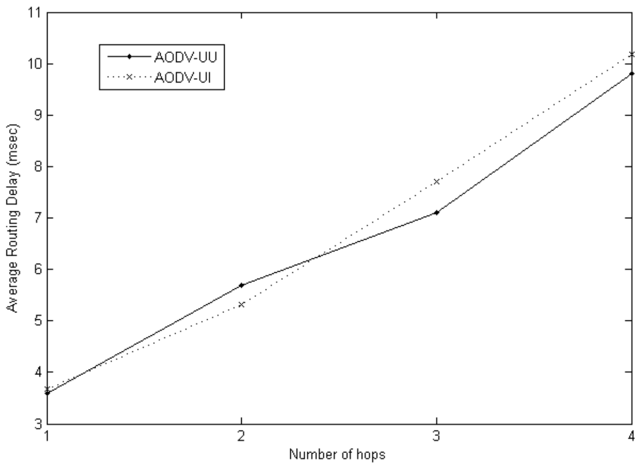

Fig. 6 Routing delay vs. number of hops on topology 1

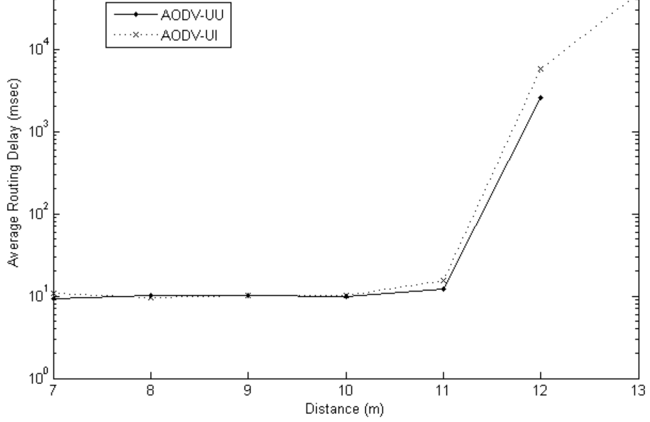

Fig. 7 Routing delays vs. inter-node distance on topology 1

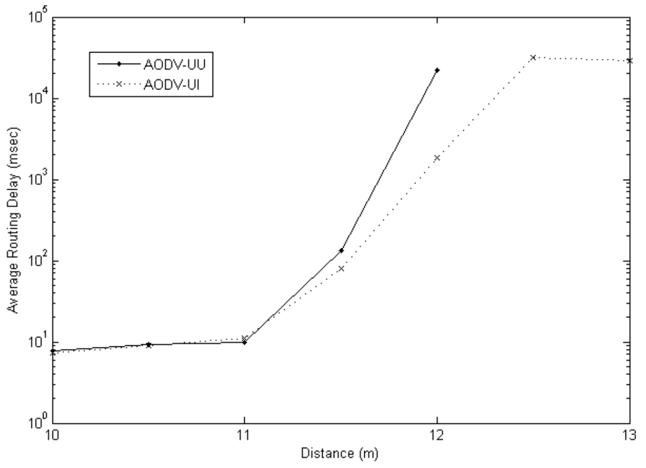

Fig. 8 Routing delay vs. inter-node distance on topology 2

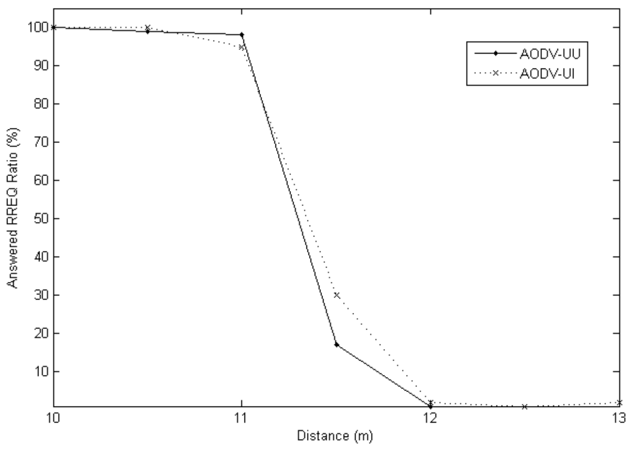

Fig. 9 Answered RREQ ratio vs. no. of hops on topology 2

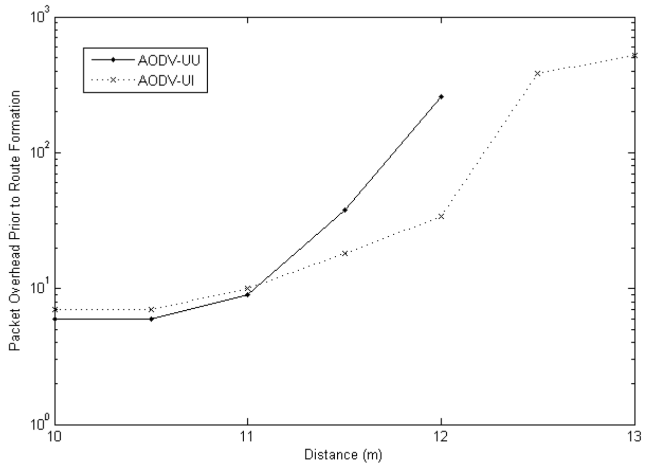

Fig. 10 Packet overhead vs. inter-node distance

On good link condition (inter-node distance <10m) both protocol performs nearly identical. But as distance growth, AODV-UI is more reliable since the replies are sent via multiple paths as seen on Fig. 8. This can be proven by examining the ratio of replies that reach the originator on Fig. 9.

Answered RREQ ratio is defined as number of RREQ/number of reply accepted at originator. In this test originator sends 100 RREQ message to the gateway and correct reply is calculated.

Ratio of both protocols drop to nearly zero on 12m, but AODV-UI manages to make few replies to reach the originator owing to multiple reverse paths.

Multiple reverse paths are an advantage at poor link quality, but it poses a burden at good link quality. Multiple reverse path means more overhead packet sent during the routing process.

Fig. 10 depicts that at better link quality AODV-UU has only 6 packets overhead compared to AODV-UI 7 packets. Note that this small difference becomes even larger with more nodes on the MANET since more nodes send redundant packets. At over 11m inter-node distance AODV-UU starts to send more packets prior to route formation. And at over 12minter-node distance the graph of AODV-UU is going to infinity since no route ever formed. This advantage surpasses the larger overhead at better link quality in harsh networking environment.

V. Concluding Remarks

The implementation of AODV-UI routing protocol on embedded hardware especially on wireless router based on MIPS architecture has been reported in this work. The implemented protocol has able to meet the expectation in the original AODV-UI simulation on NS-2. This work shows that the AODV UI which is built on top of AODV-UU has proven to perform better in a bad link quality due to the existence of the reverse route capability.

References

[1] IETF, "Ad-hoc On-Demand Distance Vector (AODV) Routing", www.ietf.org/rfc/rfc3561.txt, July 2003.

[2] Harris Simaremare, Riri Fitri Sari, "Performance Evaluation of AODV variants on DDOS, Blackhole and Malicious Attacks" IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.6, June 2011.

[3] Han-Shik Kim, Byung-Seok Kang, Sangheon Pack, Chul-Hee Kang, "Route Investigation for Secure Routing in Mobile Ad-hoc Networks," The International Conference on Emerging Security Information, Systems, and Technologies (SECURWARE), pp.163-168, 2007.

[4] Arshad, J., Azad, M.A., "Performance Evaluation of Secure on-Demand Routing Protocols for Mobile Ad-hoc Networks," 3rd Annual IEEE Communications Society on Sensor and Ad-hoc Communications and Networks (SECON '06). , vol.3, no., pp.971-975, 28-28 Sept. 2006.

[5] I.U., Jayasooriya et al. "Decentralized peer to peer web caching for Mobile Ad Hoc Networks (iCache)", International Conference on Computer Science & Education, Colombo, April 2013.

[6] Schellenberg, S. et al. "Implementation and validation of an address resolution mechanism using adaptive routing" International Conference on Information Networking (ICOIN), Bankok, Jan. 2013.

[7] Minh-Son Nguyen, Quan Le-Trung. "Integration of Atheros ath5k device driver in wireless ad-hoc router" International Conference on Advanced Technologies for Communications (ATC). Ho Chi Minh City. Oct. 2013.

[8] Abdusy Syarif, Riri Fitri Sari, "Performance Analysis of AODV-UI Routing Protocol With Energy Consumption Improvement Under Mobility Models in Hybrid Ad-hoc Network", International Journal on Computer Science and Engineering (IJCSE), Vol 3 No. 7 July 2011.

[9] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in Proc. WMCSA, New Orleans, LA, Feb. 1999, pp. 90–100.

[10] Imagination Technology, "MIPS32 Architecture", http://www.imgtec.com/mips/architectures/mips32.asp, May 2014.

[11] Rubio, Victor P. "A FPGA Implementation of a MIPS RISC Processor for Computer Architecture Education". New Mexico State University. 22 December 2011.

[12] OpenWRT, http://wiki.openwrt.org/

[13] Abdusy Syarif, Harris Simaremare, Sri Chusri Haryanti, Riri Fitri Sari, "Adding Gateway Mode for R-AODV Routing Protocol in Hybrid Ad-hoc Network " IEEE Tencon conference, Bali, 2011.

[14] Harwahyu, R. et al. "AODV-UI with Malicious Node Detection and Removal for Public MANET," Journal of Communications Software & Systems, vol. 8 no. 4, 2012.

[15] A. Hamidian, "Performance of Internet Acces Solutions in Mobile Ad-hoc Networks", Master Thesis, Lund University, 2001.

[16] C. Kim, E. Talipov, B. Ahn , "A Reverse AODV Routing Protocol in Ad-hoc Mobile Networks", IFIP-International Federation for Information Processing, Seoul, Korea, August 2006.

[17] E. Talipov, D. Jin, J. Jung, I. Ha, YJ Choi, C. Kim,"Path Hopping based on Reverse AODV for Security", APNOMS, Busan, Korea, September 2006.

[18] Rendong Bai and Mukesh Singhal, "Salvaging Route Reply for On-Demand Routing Protocols in Mobile Ad-Hoc Networks" in MSWIM 205, Montreal, Quebec, Canada. Oct 2005.

Bibliography

Boma Anantasatya Adhi receives his Bsc degree at Department of Electrical Engineering Universitas Indonesia in 2010. He receives his master study from Universitas Indonesia His interests include hi-performance computing, embedded system and intelligent control.

Ruki Harwahyu received his Bsc degree at Department of Electrical Engineering Universitas Indonesia in 2011. He received his master degree from Universitas Indonesia and National Taiwan University of Science and Technology. He has been conducting research around the topic of Internet of Things.

Abdusy Syarif receives his BSc degree in Informatic Engineering from Universitas Mercu Buana, Indonesia. And his Master degree in Electrical Engineering from Universitas Indonesia. He is currently pursuing his PhD research on ad-hoc hybrid routing protocol and wireless sensor networks.

Harris Simaremare received the B.Sc. and Master degrees in Electrical Engineering from Universitas Gadjah Mada. He is currently pursuing his PhD research on security in wireless ad-hoc network.

Riri Fitri Sari, PhD. is a Professor at Electrical Engineering Departement of Universitas Indonesia. She received her Bsc degree in Electrical Engineering from Universitas Indonesia. She receive her MSc in Computer Science and Parallel Processing from University of Sheffeld, UK. And she received her PhD in Computer Science from University of Leeds, Leeds. Riri Fitri Sari is a senior member of the Institute of Electrical and Electronic Engineers (IEEE).

Pascal Lorenz is a professor at the University of Haute-Alsace and responsible for the Network and Telecommunication Research Group. His research interests include QoS, wireless networks, and high-speed networks. He is a member of many international program committees and has served as a guest editor for a number of journals, including Telecommunications Systems, IEEE Communications Magazine, and Lecture Notes in Computer Science.