http://dx.doi.org/10.7236/JIIBC.2013.13.3.141
JIWIT 2013-1-19

고속 영상 데이터 전송을 위한 애드혹 네트워크의 테스트베드 기반 성능 분석

Performance Analysis of Ad hoc Networks on Real world Test-bed for High Speed Video Streaming

아흐메드 엠디소헬*, 윤석훈**

Ahmed Md Shohel, Seokhoon Yoon

요 약 시뮬레이션은 애드혹 네트워크 프로토콜 검증에 유용한 기법이지만 네트워킹 및 통신 환경에 대한 가정과 간소화로 인하여 실제 환경에서도 제안된 프로토콜이 요구 성능을 획득할 수 있는지 보여주기 어렵다. 따라서, 애드혹 네트워크가 현실적으로 획득할 수 있는 성능에 대한 이해를 위해서는 실제 시험 환경을 이용한 실험이 필요하며 이를 위해서는 많은 시스템 구성요소의 상호 작용을 이해하여야 한다. 본 논문에서는 건물 안에서 고속 영상 데이터 전송을 위한 실제 애드혹 네트워크 테스트베드를 구현하여 실제 네트워크 성능을 측정한다. AODV-UU를 이용하여 구축된 멀티홉 애드혹 네트워크는 Throughput, PDR, 종단간 지연 측면에서 저용량 영상데이터 전송을 위하여 요구되는 성능을 제공할 수 있음을 보인다

Abstract Simulation is an important technique to test and verify routing protocol correctness. However, simulation does not guarantee that the protocol works expectedly on a real world environment because it needs to rely on assumptions and simplified properties such as radio characteristics and effects of geographical constraints that may not reflect the real world networking environments. Therefore, a real world experiments are necessary to gain a practical insight into the actual performance of ad hoc networks. In this paper, we build a test-bed to evaluate multi-hop ad hoc networks for high speed video streaming inside a building using AODV-UU routing protocol. Our observation indicates that the considered ad hoc network can provide a sufficient performance for a low rate streaming data in terms of throughput, PDR (Packet delivery ratio) and the average delay.

Key words : Test-bed, Experiment, Ad Hoc Network, AODV-UU, Video Streaming.

Ⅰ. Introduction

Ad Hoc network is an aggregation of wireless nodes that may communicate without any per-established infrastructure. These networks are self-organizing, self-configuring and instantly deployable in response to the application needs. The transmission range of this form of networking is limited by the individual nodes

*준회원, 울산대학교 전기전자컴퓨터공학과
**정회원, 울산대학교 전기전자컴퓨터공학과 (교신저자)
접수일자 : 2013년 4월 16일, 수정완료 : 2013년 5월 25일
게재확정일자 : 2013년 6월 14일

Received: 16 April 2013 / Revised: 25 May 2013 /
Accepted: 14 June 2013
*Corresponding Author: seokhoonyoon@ulsan.ac.kr
Dept. of Electrical/Electronic and Computer Engineering

transmission ranges and is typically smaller compared to the range of cellular systems. To communicate among the nodes, a routing protocol is used to discover and setup routes between nodes. Several routing protocol have been proposed in the last two decade AODV[1], DSR[2], DSDV[3]. However most of the protocol has been evaluated and compared through simulations[5][13][14]. Since simulations are based on assumptions and simplification (e.g., radio propagation model) so experiments performed on simulator never reveals protocol critical behavior.

To avoid these modeling approximations, real world experiment is inevitable. There have been very few measurements studies on real ad hoc test-beds that can be found in the literature. Most of the available studies found are based on survey and ad hoc network design possibilities on different platform[6]. Creating an implementation of an ad hoc routing protocol is more challenging than developing a simulation. These difficulties include creating repeatable scenarios with hundreds or thousands of nodes. Creating multiple scenarios with only small variances is also challenging. An implementation needs to inter-operate with large, complex system. Some components of this system include the operating system, socket, and network interfaces. Few studies are found where experiments are confined within ping utility based test[8][9]. The Uppsala University APE test-bed[7] is one of the largest, having run tests with more than 30 nodes. The result from the APE test-bed is important and pointed out that more research in this direction is required to enrich the ad hoc networking research field.

To this aim, we report this paper our experiences of measurements conducted on a real ad hoc network to evaluate the performances of ad hoc routing protocol and comparing them in different environments, i.e. in-lab and inside building. The main contribution of our study is the performance comparison of the MANET protocol AODV using a set of real world scenarios, in our obtained results, we compare and discuss the performance of the AODV routing protocols to understand the trade-offs in the design choices for high speed video streaming. And analyze the limitations of AODV to handle high speed traffic and real world impact on performances.

The rest of the paper is organized as follows: Section II describes the overview of the AODV-UU implementation, Test-bed implementation is described in section III, Experiments setup are presented in section IV. In section V, Experimental results are reported and finally section VI concludes the paper.

II. Overview Of AODV-UU Implementation

There have been several AODV routing protocol implementations, including AODV-UU[9] (Uppsala University, Sweden), AODV-UCSB[10] (University of California, Santa Barbara), Kernel-AODV[11], AODV-UIUC[12] (University of Illinois at Urbana-Champaign) and Mad-hoc. Among them we choose AODV-UU for our test-bed. Each of the above implementation was designed and developed independently and performs the same operation.

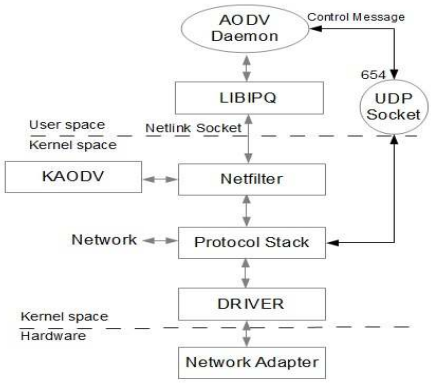


그림 1. AODV-UU 구현 구조
Fig. 1. Architecture of AODV-UU implementation.

AODV-UU is a Linux (kernel-2.4.x and 2.6.x) implementation based on RFC-3561 and runs as a user-space daemon, maintaining the kernel routing table. AODV-UU implements all mandatory and most optional feature of AODV. AODV-UU was written in

the C programming language and has been released under the General Public License (GPL). The most important feature of AODV-UU implementation is, it uses Netfilter kernel module to hack the packet into user-space which gives facility to update, modify and drop the packet in user space.

In AODV-UU implementation, kernel module components continuously listen to both inbound and outbound packets by registering itself on appropriate hooks. Packets are queued as needed to allow user-space packet processing to be performed by the AODV-UU routing daemon. Netfilter approach allows the implementation to stay independent of the kernel modification. AODV control messages generated by AODV-UU are sent on the AODV control socket (on port 654). Data packets are passes through net-link socket. Figure.1 shows the architecture of the AODV-UU implementation.

Netfilter is a Linux kernel framework for mangling packets. 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. Each packet arriving at such hook is delivered to the code segments that have registered themselves on that hook. This allows packets to be altered, dropped or re-routed by these custom code segments.

KAODV is the kernel module of AODV-UU. It registers a packet handling function on three netfilter hooks, NF_IP_PRE_ROUTING (for handling incoming packets prior to routing), NF_IP_LOCAL_OUT (for handling locally generated packets) and NF_IP_POST_ROUTING (for re-routing packets prior to sending them). Packets arriving on the NF_IP_PRE_ROUTING or NF_IP_LOCAL_OUT hook are queued in user space to allow AODV Daemon to process them. Packets are queued through LIBIPQ kernel module. LIBIPQ communicate netfilter using netlink socket and allow user space packet handling call back functions to be called whenever a queued packet arrives. Packets arriving on the NF_IP_POST_ROUTING hook are re-routed by using latest routing information available from the kernel routing table.

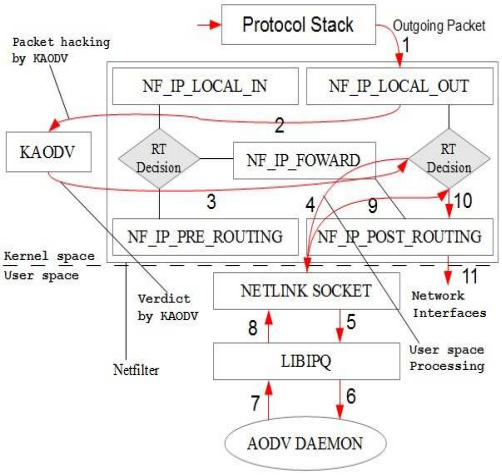


그림 2. AODV-UU와 netfilter 간의 데이터 흐름
Fig. 2. Data flow between AODV-UU and netfilter

When a packet is arrived at the protocol stack, it is hacked by respective netfilter hook and processed by registered code segment of KAODV module. A verdict is returned to netfilter. This verdict instructs netfilter to perform some packet related action, e.g., to drop the packet or to queue the packet for user space processing or to let the packet to traverse through the protocol stack.

Among the many data flow cases (e.g., AODV control packet, non IP packet), an outgoing data packet flow between netfilter and AODV-UU is shown in Fig. 2. The arrow and number denotes the sequential flow of the packet. Outgoing data packets are caught by NF_IP_LOCAL_OUT hook and send to KAODV. KAODV returns a QUEUE verdict to netfilter for queuing the packet and for further processing. If there is no valid route in the kernel routing table, packets are sent for user space processing (e.g., route discovery) by AODV daemon. After user space processing, packets are sent to network interfaces through NF_IP_POST_ROUTING.

III. Test-bed implementation

The measurement test-bed is based on an Ad hoc network made up of embedded board with same capabilities running Linux and equipped with wireless card compliant to IEEE 802.11g standard working at a constant data rate (54Mbps). All the in-lab experiments are taken place at network laboratory, and out-lab experiments are at the corridor of the 1st floor of a building. There are physical obstacles among nodes (e.g., walls) and adjacent nodes are in their respective transmission range. System components, the hardware specification and the software tools that are used to make the different experiments and network setups are descried in this section.

A. Hardware Components

Five embedded boards are used to make the different scenario of the network either in indoor and outdoor environment. All the boards are equipped with Samsung S5PV210 1GHz, ARM Cortex A8 Processor, SDRAM 256MB, ROM2 512MB NAND-Flash. This board support Linux kernel version 2.6.35.

IP-time N300UA IEEE802.11b/g/n radio adapter is used as wireless network interface card. The antennas used have Omni-directional radiation pattern. In order to setup Ad hoc Network, different nodes are all in same collision domain. Two laptops running Linux (Ubuntu 8.04.4 LTS, kernel 2.6.29.) are used to control the source and destination node.

B. Software Components

The software tools are used in this project are described in this section:

1. Operating Systems

AODV-UU implementation is based on only Linux kernel version 2.4.x and 2.6.x. Our selected embedded board supports Linux kernel version -2.6.35. So Linux kernel version 2.6.35 is used as operating system on board. After cross compiled, kernel image is uploaded to the board using zmodem.

2. AODV

Among the recently available AODV implementation, AODV-UU[4] is used as ad hoc routing Protocol. For the experiments, the latest version available, version 0.9.6 is used. This version fixes some bugs that are existed in previous version. It is based on the AODV draft version 11. AODV-UU is cross compiled with arm-generic-linux-gnueabi-gcc version 3.2 compilers.

3. Video Traffic Generator

In order to generate application level traffic in application layer, a traffic generator is developed using UDP socket. The traffic generator can generate traffic according to user given data rates. Three different ports 30000, 30001 and 30002 are defined in traffic generator script. Packet size is defined 2228 byte. Also the number of packet to send per second is calculated by script according to the user defined data rates.

On the receiving side, traffic generator creates a trace file including the information: received packet size, sequence number, time when the packet is sent, time when the packet is received at the destination host, IP address of sending node. This trace file is used to evaluate the throughput, packet delivery ratio (PDR) and average delay of the data packet. The Duration of all the experiments are 200 seconds.

4. Channel Analyzer

In order to achieve optimal reliability and throughput for a WiFi (802.11) network, it is necessary to detect and identify sources of interference that affect network performance. To detect the surrounding wireless radios, Linux IWLIST scan utility is used. Channel 1 is used as wireless channel for all of our experiments.

5. NTP Time server

Time synchronization is a challenging issue in real world implementation. All the same devices have

different clock frequency. The Network time protocol (NTP) is used to synchronize clock of all the nodes with another as reference sources. Network time Protocol is installed in a laptop which is used as a NTP server and all the nodes (embedded board) are connected to the server through Local area network (LAN). Time is synchronized on board up to less than 100 micro seconds for accuracy of the packet delay.

6. Iptables

Iptable is the user space command line program used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel. Each table contains a number of built-in chains and may also contain user-defined chains. Iptables gives us the packet filtering functionality. This is done on mac-layer. Iptables is used to ensure forced multi-hop experiment in indoor environment. The command to use for dropping packets from a specific node is:

>iptables -A INPUT -m mac –mac-source 00:26:66:4D:1A:03 -j DROP

Ⅳ. Experiments Setups

In this section, we present two different network scenarios and environments are studied. The first scenario, is in-lab environment, second scenario, is out-lab environment.

A. Scenario-1: In-lab Experiments with Single Data Flow

The reference scenario for the experiments are presented in Fig. 3. It shows a multihop network of 4 nodes. Our intention is to simulate multihop networks inside a room, where all the nodes are inside the same collision domain by taking advantage of forced multihoping. In this experiments only one node sends data at the same time.

At first, depending on the number of hops, a plan is made in which node is allowed to receive message from which node. Message from node N1 to node N4 is not allowed directly as well as from node N4 to node N1. Similarly node N1, node N3 and node N2, node N4 cannot pass message directly between them. Cross sign on arrow indicates that those messages are not allowed. Then messages come from a node that is not allowed is dropped.

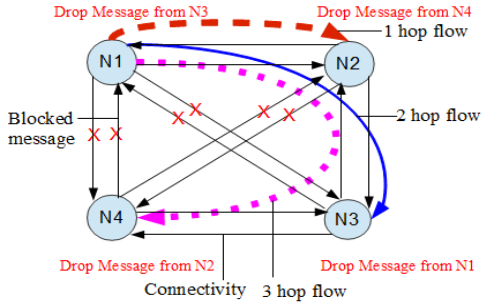


그림 3. 시나리오 1: 실험실 내부에서의 단일 플로우
Fig. 3. Scenario-1: in-lab network with single flow.

Experiments is done at room temperature and during day time. Four nodes are used for in-lab experiments and Node N1 is always the sending node and depending on the number of hop, node N2, N3 and N4 is the receiving node. All the precautions are made to avoid interference. Same set of experiments are repeated several times for producing similar results. Result analysis is presented in section V.

B. Scenario 2: In-lab Experiments with Multiple Data Flow

In the second case, the considered network topology is as Fig. 4. But this time, two data flows are produces at the same time. By default the Linux network stack is not configured for high speed large data transfer across WLAN links. This is done to save memory resources. The default value of Maximum receive socket buffer size is 128 KB in most of Linux distribution which may be enough for a low latency and general purpose network environment or applications such as DNS/web server.

Experiment is done to study the effect on

performance due to multiple flow and changed received buffer size. Buffer size is increased by multiplication of 128.

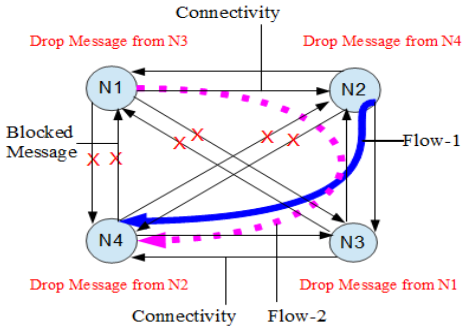


그림 4. 시나리오 2: 실험실 내에서의 다중 플로우
Fig. 4. Scenario-2: in-lab network with multiple flows

In all the experiments, two flows are started almost at the same time. Same experiment is repeated to get the same result. Node N1 and N2 work as a sending node and node N4 work as receiving node. For Convenience, data flow from node N2 to Node N4 is indicated as flow-1(2 hops) and data flow from N1 to N4 is indicated as flow-2 (3 hops). In this case there is collision between packet from Node1 and node2 which affect greatly on the performance. Result analysis is presented in section V.

C. Scenario 3: Out-lab Experiments with Multiple Data Flow

Multi-hop experiment with multiple flow are conducted at the corridor of 1st floor of the building. For all the experiment, all nodes are kept static. In Fig. 5, Node A is connected to node B but not connected to node C, D and E in terms of transmission range. Similarly node B is connected to Node A and node C but not node E. Likely Node C is connected to node D but not with node E. Node A and node C are deployed in two entry point of the building representing real time monitoring system. Node E is deployed in front of network Lab and working as receiving node. Node A and nodes C are deployed in such a way that they are 3 hops and 2 hops far from node E respectively. In this experiment, Node A and C is working as a sending node. Connectivity of two neighboring nodes are verified using PING utility before experimental traffic flows. All the nodes are set at the height of approximate 1.5 meter.

V. Experimental Results

To evaluate the performance of AODV, three performance metrics are considered for all experiments:

- Average Throughput: The amount of data received within a certain period of time for the Network.
- Packet delivery ratio: Ratio of the Number of total packet received and number of total packet sent.
- Average end-to-end delay: The time taken from when a certain packet is sent by the source node

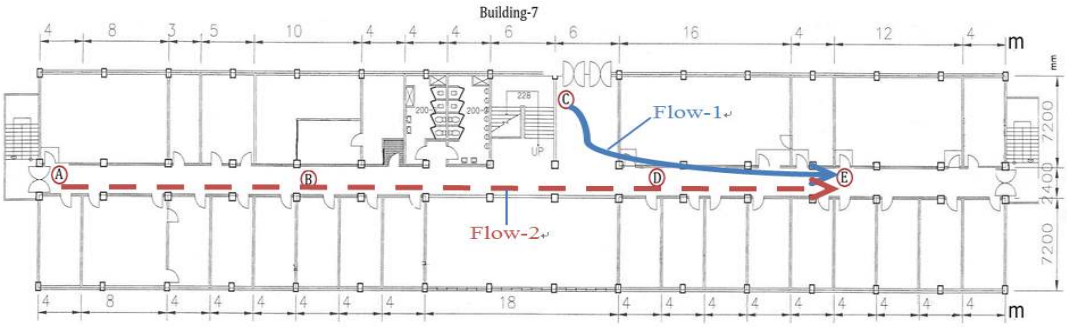


그림 5. 시나리오 3: 실험실 외부에서의 다중 플로우
Fig. 5. Scenario-3: out-lab experiments with multiple data flows.

until it is received by the destination node.

The video codec considered is H.264, which is the most powerful video compression format at the moment. It offers low bit rates with high video quality.

표 1. 프레임크기별 전송속도
Table 1. Bit rates for different frame size.

Frame size	Bit Rates(Kbps)
320x240	359
480x360	777
640x480	1363

Table shows the different bit rates for various frame size which is used for all experiments. General Parameters for all experiments are shown in Table II.

표 2. 시험을 위한 파라미터 값
Table 2. General parameter for all experiments.

Physical Layer	IEEE802.11g (54 Mbps)
Transport Layer	UDP Protocol
Network Layer	AODV-UU
Application Layer	Video Traffic Generator
Packet Size	2228 bytes
Wireless Channel	1
Wireless Mode	Ad Hoc

A. Performance Analysis: In-lab Scenario-1

In the first case, experiment is done by varying the number of hops for three different loads. In Fig. 3, first node N1 sends data to node N2 (1 hop), then node N1 to node N3 (2 hops) and finally node N1 sends to node N4 (3 hops).

In Fig. 6.(a),(b), the increase in the number of hop leads to a considerable decrease in the throughput and Packet delivery ratio for all loads. This is because new route distance initiate new fight for gaining access to medium, route discovery, maintaining connectivity and full transmission between peers. It is noticeable from PDR graph that for low speed traffic (359Kbps, 777 Kbps) PDR is reaches up to 100% for two hop distance. But for high node load (e.g., 1363 Kbps) its decreases to 73%. Which is considerably good performance for ad hoc network.

In Fig. 6(c), we see that, packet delay is increased considerably with the increase of hop distance, but packet delay for node load 1363 Kbps is drastically increases comparing to node load 359Kbps and 777 Kbps. The result of the first experiment encourage us to study the effects of received buffer size. Our 2nd experiment is varying the buffer size as well as multiple flow on AODV.

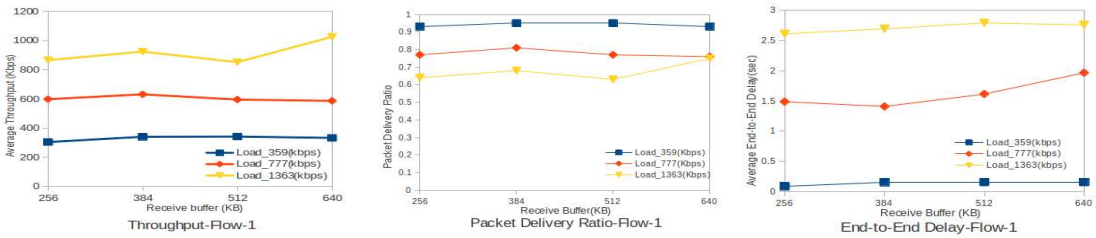
B. Performance Analysis: In-lab Scenario-2

In the second set of indoor experiments, the considered network topology is as Fig. 4, but this time, we study the effects of increased buffer size as well as multiple data flow. Fig. 7, and Fig. 8, depicts the result of flow-1 and flow-2 respectively.

The Fig. 7, (for flow-1) reveals that packet delivery ratio (PDR) as well as throughput is increased with the

(a) (b) (c)

그림 6. 시나리오 1: 실험실 내에서의 단일플로우 (a) 평균 throughut, (b) 패킷전송성공율, (c) 평균지연시간
Fig. 6. Scenario-1: in-lab experiments with single flow (a) average throughput, (b) packet delivery ratio (PDR), (c) average end to end delay.



(a) (b) (c)

그림 7. 시나리오 2: 플로우-1 (a) 평균 throughut, (b) 패킷전송성공율, (c) 평균지연시간
Fig. 7. Scenario-2: flow-1 (a) average throughput, (b) packet delivery ratio (PDR), (c) average end to end delay.

increase of receive buffer size. The increment is more significant in the case of node load 1363 (Kbps) than load 359 (Kbps) and 777 (Kbps). This is because, for high node load, number of packets generated per second is high. So, high memory is required for packet processing. As a result, throughput and PDR is increased significantly. It is noticeable that, delay also increases considerably for high load with comparing to lower load, because more packets are needed to be queued and waited for processing.

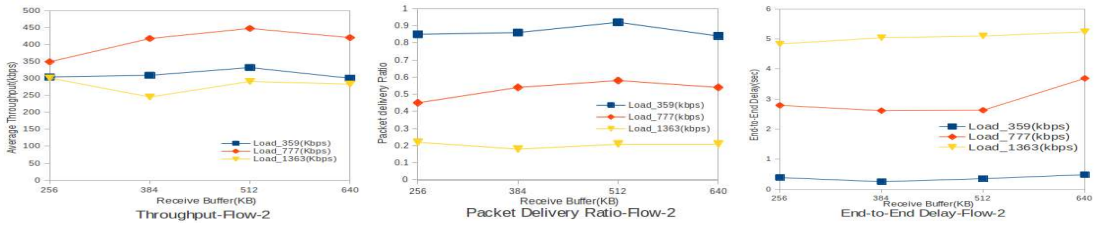
In the flow-2 (3 hops) in fig. 8(a) throughput, (b) PDR, we notice the increasing trend with the increase of buffer size. But most important outcome from this experiment is that, for higher node load (1363 Kbps), produces lower throughput, even lower than the throughput of load (359Kbps). The reason behind this is, node N2 works as sending node at the same time as a relay of node N1. In this experiment, packets from node N1 are colliding with packet from node N2. As a result a large number of packets are dropping by collision from node N1 and node N2. For high node load, large number of packets are collided which results high packets drop.

In the case of average packet delay, for flow-2, we notice that packet delay for higher node load is almost twice the packet delay of flow-1; this is because of the increase of hop distance as well as for queuing.

C. Performance Analysis: out-lab Scenario-3

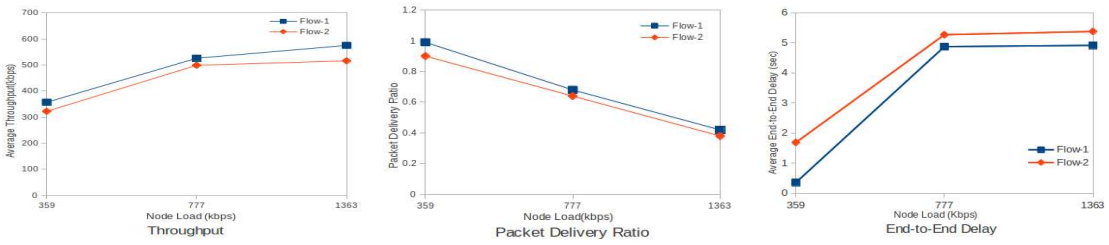
In the third set of experiments, the considered multi-hop network is depicted in Fig. 5. The characteristics of the open spaces are quite different from indoor environment. In both environments, wireless links can vary frequently. In open space, the increase of node distance makes the wireless links more unstable.

In Fig. 5, flow from node C to node E is denoted as flow-1 and flow from A to node E is denoted as flow-2. The results in Fig. 9 (a), reveals that throughput of flow-1 is significantly higher than flow-2 for all cases of the experiment. But for higher node load (1363 Kbps), both of flow-1 and flow-2



(a) (b) (c)

그림 8. 시나리오 2: 플로우-2 (a) 평균 throughut, (b) 패킷전송성공율, (c) 평균지연시간
Fig. 8. Scenario-2: flow-2 (a) average throughput, (b) packet delivery ratio (PDR), (c) average end to end delay.



(a) (b) (c)

그림 9. 시나리오 3 (a) 평균 throughut, (b) 패킷전송성공율, (c) 평균지연시간

Fig. 9. Scenario-3 (a) average throughput, (b) packet delivery ratio (PDR), (c) average end to end delay.

suffers much performance penalty.

Similarly, in the case of packet delivery ratio (PDR), for lower load (e.g., 359 Kbps and 777 Kbps) packets are delivered sufficiently (i.e., up to 99%). But for high speed traffic(e.g., 1363 Kbps) packets are dropped significantly due to the reactive nature of AODV.

We notice that all the experiment, with the increase of route distance (e.g., Flow-2) the performance of AODV decreases. In Fig. 9 (c), flow-2 (3 hops) packet delay is reaches up to 5 seconds, where flow-1 (2 hops) gets more than 4 seconds which is due to the reactive nature of AODV. Currently, we are investigating the phenomena that causes the enormous packet loss and unexpected delay as our future research.

VI. Conclusion

In this paper we have described the performances of AODV routing protocol for real ad hoc network with different scenarios and environments. In real world sense, creating a working implementation of an ad hoc routing protocol is non-trivial and more difficult than simulation. In the view of this, our paper have endeavored to describe the pros and cons of the real multi-hop ad hoc network implementation and the technique that have used with Linux kernel. The experimental results have showed that the considered ad hoc network can provide a sufficient performance for a low rate streaming data in terms of throughput, PDR (Packet delivery ratio) and the average delay.

References

[1] C. E. Perkins, E. Belding-Royer, and S. Das. "Ad hoc on-demand distance vector (AODV) routing," IETF Internet RFC 3561, July 2003.

[2] D. B. Johnson, D. A. Maltz, and Y. Hu, Dynamic source routing protocol for mobile ad hoc networks (DSR), April 2003. IETF Internet Draft, draft-ietf-manet-dsr-09.txt.

[3] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination Sequenced Distance-Vector Routing (DSDV) for mobile computers," in Proc. Of the ACM SIGCOMM'94 Conference on communication Architectures. Protocols and Applications, pp.223-244, August, 1994.

[4] E. Nordstrom. (2004) AODV implementation on Linux. AODV-UU. [Online]. Available: http://sourceforge.net/projects/aodvuu/ (last accessed on April 7, 2013).

[5] S. Das, C. E. Perkins, and M. Elizabeth. Royer. "Performance comparison of two on-demand routing protocols for ad hoc networks." Proc. of INFOCOM, pp. 3-12, March 2000.

[6] E. M. Royer, C. E Perkins, "An implementation study of the AODV routing protocol", Wireless Communications and Networking Conference, 2000, pp.1003-1008.

[7] Dept. of Computer Systems, Uppsala University (Sweden). "APE: Ad hoc Protocol Evaluation Test-bed". http://apetestbed.sourceforge.net. (last accessed on April 4, 2013).

[8] E. Borgia. "Experimental evaluation of ad hoc routing protocols." Proc. of Pervasive Computing and Communications Workshops, pp. 232-236, March 2005.

[9] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordstrm, and C. F. Tschudin, "A Large-scale Testbed for Reproducible Ad hoc protocol Evaluations," in IEEE Wireless Communications and Networking Conference 2002 (WCNC), March 2002.

[10] I. D. Chakeres, "AODV-UCSB Implementation from University of California Santa Barbara," http://moment.cs.ucsb.edu/AODV/aodv.html (last accessed on April 4, 2013).

[11] L. Klein-Berndt, "Kernel AODV from National Institute of Standard and Technology (NIST)," http://w3.antd.nist.gov/wctg/aodv_kernel/.

[12] V. Kawadia, Y. Zhang, and B. Gupta, "System Services for Implementing Ad-Hoc Routing: Architecture, Implementation and experiences," in Proceedings of MobiSys, San Francisco, CA, June 2003, pp.99-112.

[13] T. K. Song, J. J. Kang, H. S. Ahn, "A flooding Scheme Based on Packet Reception Counts for Ad Hoc Networks", The journal of the Institute of Webcasting, Internet and Telecommunication, vol.11 no.2, 2011.

[14] Y. C. Hwang, J. I. Kim, J. S. Kim, "Malicious Node Detection Mechanism of Consideration compromised Node in MANET", The journal of Korean Institute of Information Technology, vol.7 no.5, 2009.

※ 이 논문은 2011년 정부재원(교육부 기본연구사업비)으로 한국연구재단의 지원을 받아 연구되었음 (2011-0023805)

저자 소개

아흐메드 엠디소헬(준회원)

·May, 2009. B. Sc in Computer Science and Engineering, Rajshahi University of Engineering and Technology, Bangladesh.

<Interest Areas : Wireless Sensor Network, Ad hoc Network, Network Security, Embedded System>

윤 석 훈(정회원)

·2000년 2월 : 인하대학교 자동화공학과 공학사

·2005년 6월 : 뉴욕주립대 (SUNY at Buffalo) 컴퓨터공학석사

·2009년 9월 : 뉴욕주립대 (SUNY at Buffalo) 컴퓨터공학박사

·2009년 ~ 2011년 : LIG 넥스원 책임연구원

·2011년 ~ 현재 : 울산대학교 전기공학부 조교수

<관심분야 : 애드혹네트워크, 협력 로봇네트워크, 수중통신망, 전술통신망>