Implementation of AODV-UU on Linux Kernel version 3.8

Sudharsan D[1], Vamshi Raghu Nandan K[1],

Sumanth K[1], Vamsi Krishna B[1], Krishna Reddy KV[1], Raj Kumar Reddy G[1],

Revathi Venkataraman N[1], Pushpalatha M.[1]

1. Department of Computer Science and Engineering, SRM University.

ABSTRACT
Mobile Ad-hoc networks are infrastructure-less communication networks which allow participating mobile nodes to communicate without centralized control. The routing protocol “Ad-hoc On Demand Distance Vector routing protocol” (AODV)[1] is one such protocol of interest. Our goal is to make the AODV-UU[2] implementation of this protocol compatible with Linux Kernel version 3.8 as it was earlier supported only till 2.6 kernel version with a backward compatibility to 2.4 kernel version.

Keywords:
AODV[1][2], MANET[3],NETLINK[18],LINUX KERNEL[16]

1. INTRODUCTION

Ad-hoc networks are a temporary requirement based networks, where every node with a requirement of communication joins the network, it then uses other nodes as agents for forwarding data packets it has generated. MANET[4] does not have any infrastructure, hence every node has the capability to move within the range of network and have a seamless communication with any node ready for communication. Distinctive properties of MANET[4] are:

- Energy and security constraints.
- Dynamic topologies.
- Limited bandwidth.

Additionally, each node keeps track of all the nodes available in its network with the help of a routing table. This is done using any of the available routing protocols, one such routing protocol is AODV[1] routing protocol[3][4][12] which is a reactive routing protocol and updates routing table entirely based on the packets it receive when connected to the Ad hoc network which is particularly efficient for mobile nodes running on limited power sources. One of the stable and available open source implementations of AODV is AODV-UU [2] by Uppsala University, implemented on the Linux based Operating systems.

This implementation was done for the 2.6 kernel version and is not compatible with later versions of Linux kernel[16]. Our contribution towards updating is to change the variable names according to 3.8 kernel[16], resolving the driver compatibility issues when using Broadcom Wi-Fi Adapter[7]. Qualcomm Atheros Adapters have no compatibility issues with the Original version of AODV-UU. Netlink[18] socket is a special Inter-process communication scheme used for transferring information between kernel[16] and user space processes, it also provides a full-duplex communication link between the Linux kernel and user space. This is done using the standard socket API’s for user-space processes, and a special kernel API for kernel modules. Netlink sockets use the address family AF_NETLINK, as compared to AF_INET used by a TCP/IP socket.

Need for Netlink socket

- It is simple to interact with the standard Linux kernel as only a constant has to be added to the Linux kernel source code. There is no risk to pollute the kernel or to drive it in instability, since the socket can immediately be used.
- Netlink sockets are asynchronous as they provide queues, so they do not interfere with kernel scheduling.
- Netlink sockets provide the possibility of multicast.
- Netlink sockets provide a truly bidirectional communication channel: A message transfer can be initiated by either the kernel or the user space application.

The remainder of this paper is organized as follows.

We briefly describe the various implementations of AODV in Section 2.1, as well as various works done on AODV-UU implementation in Section 2.2. Section 3 describes the methodology of the process of updating. Section 4 describes the experimentation done in real-time and presents comprehensive test results from the field testing. Section 5 summarizes the work done.

2. RELATED WORK

2.1 Other Implementations of AODV

MAD-HOC [17] was the first AODV implementation, for 2.2 kernel version and it is written in Java. MAD-HOC is a metropolitan mobile ad hoc network simulator. It doesn't have queuing of data packets while route discovery is in process. Does not support multicast. Packet capturing is done using snooping.

AODV-UCSB (University of California, Santa-Barbara) [9] is an implementation for 2.4 kernel version. It uses the Netlink socket for capturing packets from kernel.

AODV-UIUC [15] implementation is similar to AODV-UCSB and AODV-UU except that it explicitly separates the routing and forwarding functions. Routing protocol logic takes place in the user-space daemon, while packet forwarding is handled in the kernel.

Kernel AODV[10] is a loadable kernel module for Linux. It implements AODV routing between computers equipped with WLAN devices. It does not need a user-space daemon as it uses Netfilter. It places the routing protocol logic inside the kernel module. This does not need packets to traverse from kernel to user-space, which improves the performance.

Proxy-AODV[13] is an implementation in which when source and destination are not connected some of the nodes called proxy nodes are selected by source to hold the data on behalf of destination. Proxy nodes act as a source and try to deliver data to the destination.

UoBWinAODV[19] is a protocol handler which uses a NDIS driver and a user space program to manage the operations of AODV. The NDIS driver is a filter driver that sits on top of the NIC driver, controlling the packet flow and obtaining information. The user space program, run on a command prompt, interacts with the NDIS filter driver using IOCTL calls and manages the routing environment as required by the AODV protocol.

AODV-BR[14] uses backup routing to provide multiple paths [5] in case of link failure by creating a mesh network.

2.2 Other Works Done in AODV-UU

Adding IPv6 compatibility to AODV-UU[20]

The IPv6 patch for AODV-UU version 0.9 aims to make the source code compilable either for IPv4 or for IPv6 networks. This is done by adding a layer for IP-version abstraction, called iplib. Within the AODV code all types and calls to the Linux kernel or socket interfaces specific to the IP-version are replaced by calls to the respective iplib types and functions.

Adding reputation extension to AODV-UU[22]

This procedure relies on the overall cooperation of nodes taking part to the Wireless Mesh Network. Each node associates a reputation value to its neighbors, which reflects the trust it puts in them. Based on that reputation value, the hop count is increased or decreased, depending whether the intermediate node is trusted or not. This mechanism enables to select the most trustworthy paths in the mesh. We present preliminary experimental results obtained with our implementation.

Implementation of Geo cast Enhanced AODV-UU in Linux Test bed[21]

This procedure uses GPS data to reduce routing delay and efficient information distribution. This Information is used to geo cast the request rather than broadcast route request in AODV-UU, thereby, reducing the number of broadcasts (routing overhead). This eventually helps to reduce the flooding of packets in the network for route discovery.

3. METHODOLOGY

The AODV-UU 0.9.6 implementation was first updated to the kernel version of 3.8, latest available version at the time of Updating, using Ubuntu 12.04.3 LTS Distribution. Most significant changes were made to the netlink socket handler functions in the code. Changes were made in the file KAODV_NETLINK.C, the functions which were changed include `NLMSG_PUT(skb, 0, 0, type, size - sizeof(*nlh))` Changed to `__NLMSG_PUT(skb, 0, 0, type, size - sizeof(*nlh))`,this change is also reflected in the method `kaodv_netlink_build_msg` where it is called, variables in the method `void kaodv_netlink_send_rt_update_msg` concerned with `kaodv_rt_msg` type are changed,

The protocol had issues with the Broadcom Wi-Fi adapter so a BCML kernel source is added explicitly for smooth functionality of protocol.

The field testing[8][11] was carried out at the open air auditorium inside university campus. We have used laptops pre-Installed with Ubuntu OS, and python testbed which helps with running AODV-UU, other packages used are:

- Tshark-network analyzer from wireshark.
- WxPython- wrapper for GUI.
- PythonNumpy-package for scientific computing.
- Vsftpd-an FTP server.
- Ssh- cryptographic network protocol for secure data communication.
- Hping3- network tool able to send custom TCP/IP packets and to display target replies like ping. program does with ICMP replies.
- Bison- a parser generator for context-free languages.
- Flex- a tool for generating scanner.

Laptops were acting as nodes and every node was moving dynamically in random directions to simulate a dynamic topology. The python test bed enables nodes to select files from the file system and transfer them to any node which is within the network. If the node is not in direct communication, then an intermediate neighbor is used as transmitter of our data. This process is transitive, hence every node is accessible in network, if not directly, at least through any intermediate nodes participating in the network.

The experimentation also produces log folders containing 3 files. Namely, aodv.txt containing the log information from for AODV routing, aodv-rt.txt contains the routing table information, aodvoutput containing logs from the tshark command which were used to analyze the network.

4. EXPERIMENTATION AND TEST RESULTS

Methodology described in the previous section is presented in the graphs packet loss ratio, delay in packet transfers, throughput of the network is shown in the network analysis graphs.

4.1 Packet Loss

An important parameter in analyzing the network is packet loss ratio. Packet when arrived in a network layer is forwarded when a valid route to the destination is known, otherwise it is buffered until a valid route is found. A packet is lost when the buffer is full or the time that the packet has been buffered, exceeds the limit.

PLoss = DataSent – DataRec

This loss denotes the number of packets which are lost either due to buffer overflow or it has taken more time than the Time to Live (TTL) value set by the sender.

The packet loss ratio from the figure 1 shows the way in which packets are lost in the network during in the initial phase of testing. The ratio gradually starts decreasing after initial phase and finally attains a constant value of 0.5 after 400 msecs.

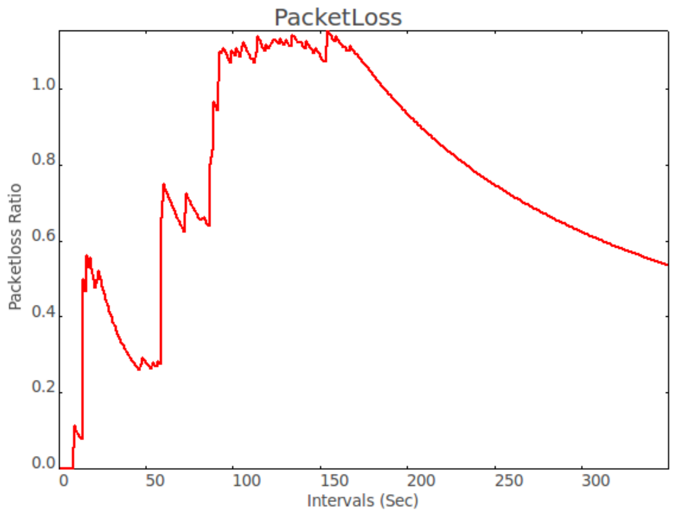

Figure 1

4.2 Delay

Delay parameter measured for a network is the latency used by a packet to navigate from its originator to destination.

The traffic type used is UDP. Initially, the graph shows high delay ratio and gradually decrease to finally stabilize at 0.13.

Two massive sources contributing to packet loss are retransmissions due to Ad hoc networks and Wireless LAN MAC layers. Figure 2 shows the delay ratio of the network. It initially went high and gradually stabilized at .125 secs.

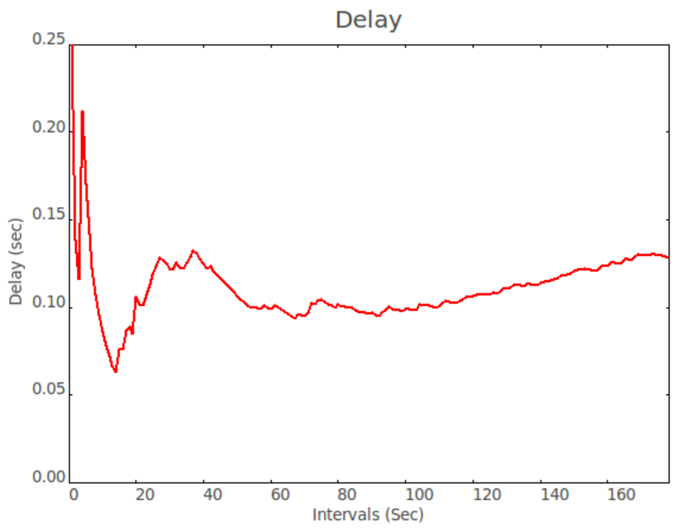

Figure 2

4.3 Throughput

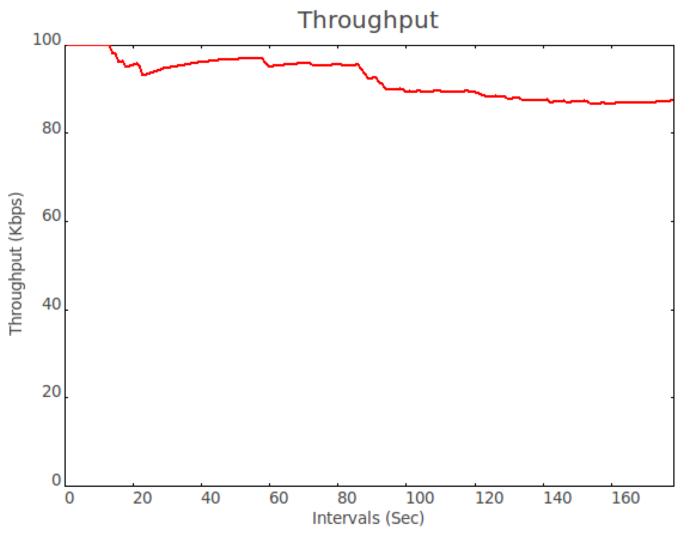

Figure 3

In communication networks, such as Ethernet or packet radio, throughput or network throughput is the rate of successful message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot. The system throughput or aggregate throughput is the sum of the data rates that are delivered to all terminals in a network. Throughput refers to how much data can be transferred from one location to another in a given amount of time. Figure 3 describes the throughput achieved during the experimentation. It has decreased over time and finally stabilized at 90 Kbps.

5. Conclusion

The Experimental results establish that AODV-UU Protocol is working well with the 3.8 Linux Kernel version. The network parameters analyzed were well in accordance with all the parameters of the results from previous works done in AODV-UU[6].

Acknowledgment

This work is funded in part by the Defence Research and Development Organization under grant no: ERIP/ER/1203090/M/01/1474.

References

1. C. Perkins, E. Belding-Royer, S. Das, RFC Draft for experimental AODV. https://www.ietf.org/rfc/rfc3561.txt.
2. AODV-UU source from GitHub. https://github.com/erimatnor/aodv-uu
3. Klein-Berndt L., A quick guide to AODV routing, National Institute of Standards and Technology, (2001).
4. Perkins C., Belding-Royer E. and Das S., Ad hoc on demand distance vector (AODV) routing (RFC 3561), IETF MANET Working Group (2003).
5. Marina M.K. and Das S.R., On-demand multipath distance vector routing in ad hoc networks, IEEE, (2001).
6. Hua Yang, Zhi-yuan Li, *Simulation and analysis of a modified AODV routing protocol*, Computer science and network technology (ICCSNT), 2011.
7. IEEE Computer Society , IEEE 802.11 Standard , IEEE standard for Information Technology, 1999.
8. V. kawadia, Y. Zhang and B. Gupta. System Services for Implementing Ad-Hoc Routing: Architecture, Implementation and Experiences. In *Proceedings of the 1st International conference on Mobile Systems, Applications, and Services (MobiSys), Pages 99-112, San Francisco ,CA, June 2003.*
9. I.D. Chakeres. AODV-UCSB Implemetation from University of California Santa Barbara. http://moment.ucsb.edu/AODV/aodv.html.
10. L. Klein-Berndt. Kernel AODV from National Institute of standards and Technology (NIST) http://w3.antd.nist.gov/wctg/aodv_kernel/
11. D.A. Maltz, J. Broch, and D.B. Johnson, *Experiences Designing and Building a Multi-hop Wireless Ad hoc Testbed*. Technical Report CMU. CMU school of Computer science.
12. E.M. Royer and C.E. Perkins . Multicast Operation of the Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol . *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), pages 207-218, Seattle, WA, August 1999.*
13. Anshuman Tiwari and Sridhar Iyer , *proxy-AODV : Extension of AODV For Partially Connected Ad hoc Networks*.
14. Sung-Ju Lee and Mario Gerla, *AODV-BR Backup Routing in Ad hoc Networks.* Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE , Volume: 3, Digital Object Identifier: 10.1109/WCNC.2000.904822 Publication Year: 2000 , Page(s): 1311 - 1316 .
15. AODV-UIUC University of Illinois at Urbana Champaign http://sourceforge.net/projects/aslib/files/AODV-UIUC/
16. Ubuntu release notes for 12.04 LTS version on 3.8 Linux Kernel, https://wiki.ubuntu.com/PrecisePangolin/ReleaseNotes/UbuntuDesktop/UbuntuDesktop-12.04
17. F.Lilieblad ,O.Mattson, P.Nylund,D.Ouchterlony, and A.Roxenhag. Mad-hoc AODV Implementation and Documentation. http://mad-hoc.flyinglinux.net,
18. Pablo Neira Ayuso, Rafael M. Gasca and Laurent Lefevre, "Communicating between the kernel and user space in Linux using Netlink sockets"
19. Koojana Kuladinithi, *Release of UoBWinAODV - An AODV Protocol Handler for Windows.* http://www.ietf.org/mail-archive/web/manet/current/msg05460.html
20. IPv6 compatibility for AODV. http://www.ietf.org/mail-archive/web/manet/current/msg05795.html
21. Mubarik, M.A., Khan, S.A. , Hassan, S.A. , Sarfraz, N. Implementation of geocast enhanced aodv-uu in linux testbed. June 2009.
22. Guillaume, Julien "Adding reputation extensions to AODV-UU".

Sudharsan D. Junior Research Fellow, Department of Computer Science and Engineering, SRM University.

Vamshi Raghu Nandan K, Student, Department of Computer Science and Engineering .