

A O D V

Ad hoc On Demand Distance Vector Routing Protocol

Introduction

In November 2001 the MANET (Mobile Ad-hoc Networks) Working Group for routing of the IETF community has published the first version of the AODV Routing Protocol (Ad hoc On Demand Distance Vector).

AODV belongs to the class of Distance Vector Routing Protocols (DV). In a DV every node knows its neighbours and the costs to reach them. A node maintains its own routing table, storing all nodes in the network, the distance and the next hop to them. If a node is not reachable the distance to it is set to infinity. Every node sends its neighbours periodically its whole routing table. So they can check if there is a useful route to another node using this neighbour as next hop. When a link breaks a Count-To-Infinity could happen.

AODV is an 'on demand routing protocol' with small delay. That means that routes are only established when needed to reduce traffic overhead. AODV supports Unicast, Broadcast and Multicast without any further protocols. The Count-To-Infinity and loop problem is solved with sequence numbers and the registration of the costs. In AODV every hop has the constant cost of one. The routes age very quickly in order to accommodate the movement of the mobile nodes. Link breakages can locally be repaired very efficiently. To characterize the AODV with the five criteria used by Keshav AODV is distributed, hop-by-hop, deterministic, single path and state dependent.

AODV uses IP in a special way. It treats an IP address just as a unique identifier. This can easily be done with setting the Subnetmask to 255.255.255.255. But also aggregated networks are supported. They are implemented as subnets. Only one router in each of them is responsible to operate the AODV for the whole subnet and serves as a default gateway. It has to maintain a sequence number for the whole subnet and to forward every package.

In AODV the routing table is expanded by a sequence number to every destination and by time to live for every entry. It is also expanded by routing flags, the interface, a list of precursors and for outdated routes the last hop count is stored.

Unicast Routing

For unicast routing three control messages are used: RREQ (Route REPLY), RREP (Route REPLY), RERR (Route ERRor). If a node wants to send a packet to a node for which no route is available it broadcasts a RREQ to find one. A RREP includes a unique identifier, the destination IP address and sequence number, the source IP address and sequence number as well as a hop count initialised with zero and some flags. If a node receives a RREQ which it does not have seen before it sets up a reverse route to the sender. If it does not know a route to the destination it rebroadcasts the updated RREQ especially incrementing the hop count. If it knows a route to the destination it creates a RREP.

The RREP is unicasted to the origin node taking advantage of the reverse routes. A RREP contains the destination IP address and sequence number, the source IP address, a time to life, a hop count as well as a prefix only used for subnets and some flags. When a node receives a RREP it checks if the hop count in the RREP for the emitter of the message is lower than the one in its own routing table or the destination sequence number in the message is higher than the one in its own routing table. If none of them is true it just throws the package away. Otherwise it updates its routing table and if it is not the destination it reunicasts the RREP.

In mobile network link breakage is very common. If a node realises that other nodes are not any longer reachable it broadcasts a RERR containing a list of the unreachable nodes with their IP addresses and

sequence number and some flags. A node who receives a RERR iterates over the list of unreachable destinations checking if a next hop in its routing table contains one of these nodes. If yes it updates its routing table. If the receiving node still maintains routes to unreachable nodes it broadcasts its own RERR containing this information.

Routes and link lifetime are extended by sending a package over it and by hello messages. A hello is a special RERR which is only valid for its neighbours. A node may broadcast periodically a hello message so that no link breakages are assumed by its neighbours when they do not hear anything from it for a long time.

If a link in an active route breaks a node can try to repair the route locally. To do this, it releases a RREQ to find a new route to the destination on the broken link side not touching the other direction of the route.

It exists another special package a RREP-ACK which is used for unreliable or unidirectional links. Also some other special mechanisms are used like precursors to track the list of active routes for using in RERR emission.

Multicast Routing

One of the great advantages of AODV is its integrated multicast routing. In a multicast routing table the IP address and the sequence number of the group are stored. Also the leader's IP address and the hop count to him are stored as well as the next hop in the multicasting tree and the lifetime of it. To join a multicast group a node has to send an RREQ to the group address with the join flag set. Any node in the multicast tree which receives the RREQ can answer with a RREP. Like this a requester could receive several RREPs from which he can choose the one with the shortest distance to the group. A MACT (Multicast Activation) Message is sent to the chosen tree node to activate this branch. If a requester does not receive a RREP, the node supposes that there exists no multicast tree for this group in this network segment and it becomes the group leader. A multicast RREP contains additionally the IP of the group leader and the hop count to the next group member. The group leader broadcasts periodically a group hello message (a RREP) and increments each time the sequence number of the group.

When two network segments become connected, two partitioned group trees have to be connected. Every group member receiving two group hello messages from different leaders will detect a tree connection. Then this node emits an RREQ with the repair flag set to the group. If a node in the group tree does not receive any group hello or other group message it has to repair the group tree with a RREQ and has to ensure that not a RREP from a node in its own subtree is chosen. If a group member wants to leave the group and it is a leaf it can prune the branch with a MACT and the flag prune set. If it is not a leaf it must continue to serve as a tree member.

Security

Security is a very dangerous point in mobile communication. AODV defines no special security mechanisms. So an impersonation attack can easily be done. In order to prevent this an authentication is required for example with PKA (public key authentication). Messages can easily be intercepted. In order to prevent this you can cipher them for example with a PK (public key). Standard IP security protocols like IPsec should not be used.

Implementations

There are two types of different implementations, user space daemons and kernel modules. The first of them requires to maintain an own routing table and was first implemented in the Mad-hoc Implementation by Fredrik Lilieblad, Oskar Mattsson, Petra Nylund, Dan Ouchterlony, Anders Roxenahg running on a Linux 2.2 kernel but does not support multicast. The University of Uppsala also published an user space daemon implementation called AODV-UU which runs on Linux with a 2.4 kernel. The newest daemon Momnet was published on the 2nd of April 2002 by the University of California, Santa Barbara also running on a Linux with a 2.4 kernel. The only kernel implementation was done by the

NIST, Department of Commerce's Technology Administration U.S., Wireless Communications Technologies Group running on a Linux with a 2.4 kernel. It is very fast and efficient reaching the best performance of all implementations.

References:

An Engineering Approach to Computer Network, S. Keshav, Addison Wesley / AT&T, 1997

Mobile Ad Hoc Networking Working Group - AODV

<http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-10.txt>

AODV Multicast Features

http://www.tml.hut.fi/Opinnot/Tik-110.551/2000/papers/AODV_features/

AODV-UU routing protocol implementation created at Uppsala University

<http://www.docs.uu.se/~henrikl/aodv/>

NIST Linux AODV

http://w3.antd.nist.gov/wctg/aodv_kernel/

<http://w3.antd.nist.gov/wctg/manet/AODVReadme.pdf>

IETF Manet Working Group AODV Draft

<http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt>

AODV Short

<http://www.cs.ucsb.edu/~eroyer/aodv.html>

AD-HOC NETWORKS

<http://www.cmpe.boun.edu.tr/~emre/research/mstthesis/node27.html>

NEC ResearchIndex

<http://citeseer.nj.nec.com/context/485757/0>

AODV code for CMU Wireless and Mobility Extensions to ns-2

<http://www.ececs.uc.edu/~mmarina/aodv/>

HUT AODV for IPv6

<http://www.tml.hut.fi/~ajtuomin/manet/aodv/>

FlyingLinux / MAD-HOC / Technical Documentation

<http://mad-hoc.flyinglinux.net/techdoc.ps>

Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks

http://www.cs.ucsb.edu/~eroyer/txt/aodv_infocom.ps

5th Annual ACM/IEEE International Conference on Mobile Computing and Networks (MOBICOM)

http://www.cs.ucsb.edu/~eroyer/txt/aodv_mobicom99.pdf

2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)

http://www.cs.ucsb.edu/~eroyer/txt/aodv_WMCSA_slides.ps

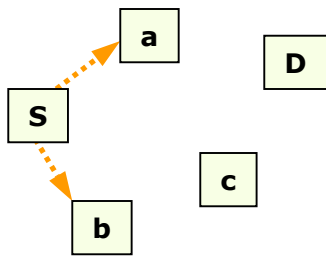
An Implementation Study of the AODV Routing Protocol

http://www.cs.ucsb.edu/~eroyer/txt/wcnc_impl.ps

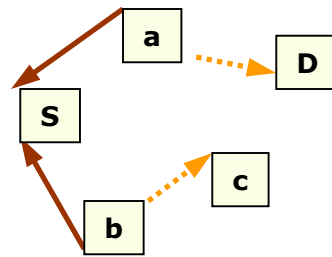
Momnet, University of California, Santa Barbara

<http://moment.cs.ucsb.edu/AODV/aodv-ucsb-0.1.tar.gz>

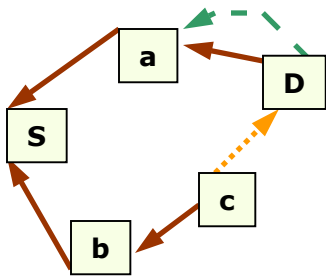
AODV – Route Establishment



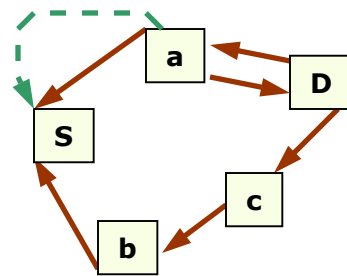
(1) S wants to send a packet to D
S broadcasts an RREQ



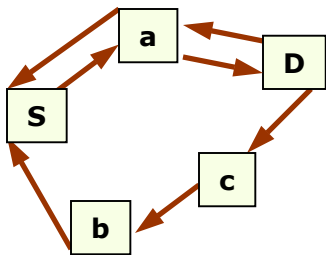
(2) a & b establish Reverse Route
a & b rebroadcast RREQ



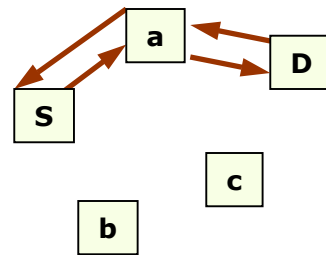
(3) c & D establish Reverse Route
c rebroadcasts RREQ
D unicasts RREP



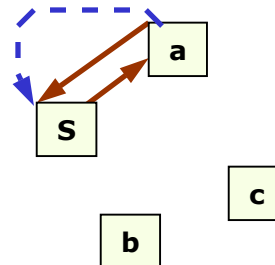
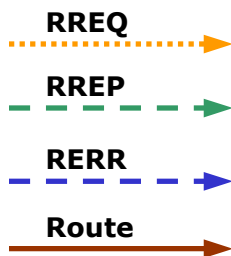
(4) D establishes Reverse Route
D drops duplicate RREQ
a establishes Route
a unicasts RREP



(5) S establishes Route

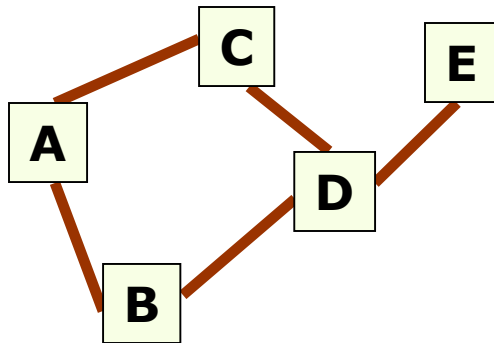


(6) Unused reverse routes expire



(7) Link between a and D broken
a unicasts RERR

DV Routing Protocol



Distance Vector Routing Protocol Initialisation Phase

- Cost per hop = 1

Each node:

- Knows its neighbours and the cost to reach them
- Tells its neighbours periodically the distance to every other node in the network

Route Table A

DE	CO	NH
A	0	A
B	1	B
C	1	C
D	∞	-
E	∞	-

Route Table B

DE	CO	NH
A	1	A
B	0	B
C	∞	-
D	1	D
E	∞	-

Route Table C

DE	CO	NH
A	1	A
B	∞	-
C	0	C
D	1	D
E	∞	-

Route Table A

DE	CO	NH
A	0	A
B	1	B
C	1	C
D	2	B
E	∞	-

Route Table B

DE	CO	NH
A	1	A
B	0	B
C	2	D
D	1	D
E	2	D

Route Table C

DE	CO	NH
A	1	A
B	2	D
C	0	C
D	1	D
E	2	D

Route Table A

DE	CO	NH
A	0	A
B	1	B
C	1	C
D	2	B
E	3	B

DE - Destination
CO - Cost
NH - Next Hop

AODV

Ad hoc On demand Distance Vector

Routing Protocol

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Overview

- Introduction
- Characteristics
- The protocol
- Special packages and mechanism
- Multicasting (if we have enough time)
- Security
- Implementations

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – History

- IETF community the MANET
- November 2001 published
- AODV Routing Protocol (Ad hoc On demand Distance Vector)

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – DV

AODV a DV (Distance Vector Routing Protocol)

Characteristics of a DV Routing Protocol

- Knowledge of neighbours
- Knowledge of cost reaching each neighbour

Idea:

- Each node tells its neighbours periodically the distance to every other node in the network
- Every node maintains a routing table with all known nodes, next hop and costs
- Problems: Count-To-Infinity

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Characteristics

- On demand (with small delay)
- Unicast / Multicast / Broadcast provided
- Loop free
- Quick aging
- Link breakages efficiently repaired

- Distributed Routing
- Hop-by-hop
- Deterministic
- Single path
- State-dependent

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – IP Based

- IP as unique identifier
- No routing based on network part (*Subnet 255.255.255.255*)

Aggregated Networks (Subnets)

- Subnet router, operates AODV for the whole subnet
- Single destination sequence for whole subnet

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Route Table

- Destination IP
- Destination Sequence Number
- Hop Count to Destination (cost per hope = 1)
- Next Hop
- Lifetime

- Last Hop Count
- Routing Flags
- Interface (i.e. eth0, eth1)
- List of Precursors

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Creating RREQ

RREQ – Route REQuest Packet

- Source node wants to send a packet to a node for which no route is available
- Source broadcast RREQ with
 - RREQ ID (unique)
 - Destination IP & Sequence Number
 - Source IP & Sequence Number
 - Hop Count = 0
 - Flags

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Receiving RREQ

IF not seen before

- Set up reverse route to node where RREQ came from

IF route to destination unknown

- Rebroadcast RREQ
- Inc hop count

IF route to destination known or node is destination

- Create RREP

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Creating RREP

RREP – Route REPLY Packet

- Destination unicasts RREP to the node that relayed the RREQ
 - Destination IP & sequence number
 - Source IP
 - Lifetime
 - Hop Count = 0
 - Prefix Size = 0 (used for Subnets)
 - Flags

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Receiving RREP

IF hop count: RREP < rout table
OR IF destination sequence: route table < RREP

- Update route table

IF node is not destination

- Reunicast RREP (use reverse route)
- Inc hop count

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Creating RERR

RREP – Route ERROR Packet

- Link break or host unreachable
- Node broadcasts RERR
 - Number of unreachable destination
 - Unreachable IPs & sequence numbers
 - Flags

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Receiving RERR

FOR all unreachable destinations

IF next hop to unreachable destination is source of RERR

- Update route table

IF routes to unreachable destinations exists

- Broadcast RERR

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Route Management

- Whenever a route is used extend lifetime

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Sequence Number

-> Loop free

Own sequence number incremented

- Before originating a RREQ
- Before originating a RREP

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Hello

- A RREP
- TTL = 1 (RREP only valid for neighbours)
- Hop Count = 0

- Periodically send from every node
- If a node does not receive any messages from a neighbour (regular packet, control packet or hello) it assumes that the link is broken

- Hellos are not required but recommended

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

● AODV – Local Repair

IF a node in an active route loses connection to its neighbour it can

- Repair the route locally by releasing a RREQ to find a new route to the destination

OR

- Release a RERR

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

● AODV – More Special Packets

- RREP-ACK (for unreliable or unidirectional links)

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – More

- Precursors (list of active routes)

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Multicast Route Table

- Multicast Group IP Adress
- Multicast Group Leader IP Address
- Multicast Group Sequence Number
- Hop Count to Multicast Group Leader
- Next Hops
- Lifetime

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Multicast Group Join

- To join a Multicast Group send a RREQ with the join flag set to the address of the group
- Any node in the Multicast Tree which receives the RREQ can answer with a RREP
- Several RREP can be received by a node
- A MACT (Multicast ACTivation) message is sent to the best Route to activate it
- If no RREP is received the source node becomes the group leader

- Multicast RREP contains
 - Group Leader IP
 - Hop Count to next group member

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Multicast Group Maintains 1

- The group leader broadcasts periodical a Group Hello message (a RREP, the group sequence number is incremented for every Group Hello message)
- Reconnecting two partitioned trees (after a disconnected networks gets connected again)
 - Every group member receiving two Group Hello messages with two different group leader addresses detects the reconnection
 - This node unicasts an RREQ (flag = repair)

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Multicast Group Maintains 2

- If a node in the Group Tree does not receive any Group Hello or other Group Message it has to repair the Group Tree (with a RREQ and has to ensure that not a RREP from a node in its own group sub tree is chosen)
- A group member wants to leave the group
 - IF it is a leaf it can prune the branch with a MACT (flag = prune)
 - IF it is not a leaf it must continue to serve as a Tree Member

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Security

- Message encoding i.e. PK or trust all nodes on route
- Authentification i.e. PKA (impersonation attacks)
- No IPsec should be used

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Implementations 1

User Space Daemon

- IP Routing
- Second AODV route table in daemon

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Implementations 2

User Space Daemon

Mad-hoc Implementation

- Runs on Linux 2.2 kernels
- No multicasting

• AODV-UU by Uppsala University

- Runs on Linux 2.4.x kernels
- Full implementation

• Momnet by University of California

- Runs on Linux 2.4.x kernels with netfilter compiled in
- Full implementation NEW 2nd April 2002

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net

AODV – Implementations 3

Kernel

NIST Linux AODV by NIST

- Runs on Linux 2.4.x kernels
- Loadable kernel module
- Full implementation

- Faster and more efficient
- Kernel is changed

AODV, Presentation at ETH Zürich April 02
© Rainer Baumann, baumann@hypert.net

hypert.net