An Overview of AODV Routing Protocol

Prashant Kumar Maurya[1], Gaurav Sharma[2], Vaishali Sahu[3], Ashish Roberts[4], Mahendra Srivastava[5]
[1, 2, 4, 5] M.Tech Scholar SSET, SHIATS ALLAHABAD
[3] M.Tech Scholar MNNIT ALLAHABAD

Abstract: The Ad-hoc On-Demand Distance Vector (AODV) routing protocol is designed for use in ad-hoc mobile networks. AODV is a reactive protocol: the routes are created only when they are needed. It uses traditional routing tables, one entry per destination, and sequence numbers to determine whether routing information is up-to-date and to prevent routing loops.

An important feature of AODV is the maintenance of time-based states in each node: a routing-entry not recently used is expired. In case of a route is broken the neighbors can be notified.

Route discovery is based on query and reply cycles, and route information is stored in all intermediate nodes along the route in the form of route table entries. The following control packets are used: routing request message (RREQ) is broadcasted by a node requiring a route to another node, routing reply message (RREP) is unicasted back to the source of RREQ, and route error message (RERR) is sent to notify other nodes of the loss of the link. HELLO messages are used for detecting and monitoring links to neighbors.

I. INTRODUCTION

Wireless communication technology is steadily and rapidly increasing. People wish to use their network terminals (laptops, PDAs, etc.) anywhere and anytime. Wireless connectivity gives users the freedom to move where they desire.

There exist numerous different wireless networks varying in the way the nodes interconnect. They can be classified in two main types: Networks with fixed infrastructure and Ad hoc wireless networks

Typical for networks with fixed infrastructure is using of access points. An access point (AP) can act as a router in the network, or as a bridge. Examples for this type of networks are GSM and UMTS cellular networks [1]. APs have more information about the network and are able to route the packets the best way. In contrast, ad hoc networks have no fixed infrastructure or administrative support, the topology of the network changes dynamically as mobile nodes joins or leaves the network. In ad-hoc wireless networks the nodes them-selves use each other as routers, so these nodes should be more intelligent than the nodes in centralized networks with APs. There are a lot of situations where ad hoc networks are needed: military operations, emergency services, conferencing, game parties, home networking, etc.

If the wireless nodes are within the range of each other, the routing is not necessary. If a node moves out of this range, and they are not able to communicate with each other directly, intermediate nodes are needed to organize the network which takes care of the data transmission. The purpose of a routing algorithm is to define a scheme for transferring a packet from one node to another. This algorithm should choose some criteria to make rout-ing decisions, for instance number of hops, latency, transmission power, bandwidth, etc.

The topology of mobile ad hoc networks is time-varying, so traditional routing techniques used in fixed networks cannot be directly applied here. There are various techniques for tracking changes in the network topology and re-discovering new routes when older ones break. Since ad hoc networks have no infrastructure these operations should be performed with collective cooperation of all nodes.

Routing protocols in mobile networks are subdivided into two basic classes [12]. Proactive routing protocols (e.g. OLSR) are table-driven. They usually use link-state routing algorithms flooding the link information. Link-state algorithms maintain a full or partial copy of the network topology and costs for all known links. The reactive routing protocols (e.g. AODV) create and maintain routes only if these are needed, on demand. They usually use distance-vector routing algorithms that keep only information about next hops to adjacent neighbors and costs for paths to all known destinations. Thus, link-state routing algorithms are more reliable, less bandwidth-intensive, but also more complex and compute- and memory-intensive.

In on-demand routing protocols a fundamental requirement for connectivity is to discover routes to a node via flooding of request messages. The AODV routing protocol [2–4] is one of several published reactive routing protocols for mobile ad-hoc networks, and is currently extensively researched.

The rest of the paper is organized as follows. In the next section the AODV protocol is briefly reviewed. The properties of AODV and comparison between AODV and OLSR will be considered in Section III. Section IV will conclude this paper.

II. AD-HOC ON-DEMAND VECTOR

AODV is a relative of the Bellman-Ford distant vector algorithm, but is adapted to work in a mobile environment. AODV determines a route to a destination only when a node wants to send a packet to that destination. Routes are maintained as long as they are needed by the source. Sequence numbers ensure the freshness of routes and guarantee the loop-free routing.

Routing tables

Each routing table entry contains the following information [2] as destination, next hop, number of hops, destination sequence number, and active neighbors for this route and expiration time for this route table entry. Expiration time, also called lifetime, is reset each time the route has been used. The new expiration time is the sum of the current time and a parameter called active route timeout. This parameter, also called route caching timeout, is the time after which the route is considered as invalid, and so the nodes not lying on the route determined by RREPs delete their reverse entries. If active route timeout is big enough route repairs will maintain routes. RFC 3561 defines it to 3 seconds.

Control messages

Routing request

When a route is not available for the destination, a route request packet (RREQ) is flooded throughout the network. The RREQ contains the following fields [3]:

Source address	Request ID	Source sequence No.	Destination Address	destination sequence No.	Hop count

The request ID is incremented each time the source node sends a new RREQ, so the pair (source address, request ID) identifies a RREQ uniquely. On receiving a RREQ message each node checks the source address and the request ID. If the node has already received a RREQ with the same pair of parameters the new RREQ packet will be discarded. Otherwise the RREQ will be either forwarded (broadcast) or replied (unicast) with a RREP message: if the node has no route entry for the destination, or it has one but this is no more an up-to-date route, the RREQ will be rebroadcasted with incremented hop count and if the node has a route with a sequence number greater than or equal to that of RREQ, a RREP message will be generated and sent back to the source. The number of RREQ messages that a node can send per second is limited.
There is an optimization of AODV using an expanding ring (ESR) technique when flooding RREQ messages [5, 6]. Every RREQ carries a time to live (TTL) value that specifies the number of times this message should be re-broadcasted. This value is set to a predefined value at the first transmission and increased at retransmissions. Retransmissions occur if no replies are received. Historically such floodings used a TTL large enough - larger than the diameter of the network - to reach all nodes in the network, and so to guarantee successful route discovery in only one round of flooding. However, this low delay time approach causes high overhead and unnecessary broadcast messages. Later, it was shown [7, 8] that the minimal cost flooding search problem can be solved via a sequence of flooding with an optimally chosen set of TTLs.

Routing reply

If a node is the destination, or has a valid route to the destination, it unicasts a route reply message (RREP) back to the source. This message has the following format

Source Address	destination Address	destination sequence No.	hop count	life-Time

The reason one can unicast RREP back is that every node forwarding a RREQ message caches a route back to the source node (see section 2.4.1).

Route error

All nodes monitor their own neighbourhood. When a node in an active route gets lost, a route error message (RERR) is generated to notify the other nodes on both sides of the link of the loss of this link.

HELLO messages

Each node can get to know its neighbourhood by using local broadcasts, so-called HELLO messages. Nodes neighbors are all the nodes that it can directly communicate with. Al-though AODV is a reactive protocol it uses these periodic HELLO messages to inform the neighbors that the link is still alive. The HELLO messages will never be forwarded because they are broadcasted with TTL = 1. When a node receives a HELLO message it refreshes the corresponding lifetime of the neighbour information in the routing table.

This local connectivity management should be distinguished from general topology management to optimize response time to local changes in the network.

Sequence numbers

Counting to infinity

The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path - as Tanenbaum [9] notes. So if Y detects a link to Z is broken, but X still has a ”valid” path to Z, Y assumes X in fact does have a path to Z. So X and Y will start updating each other in a loop, and the problem named ”counting to infinity” arises. AODV avoids this problem by using sequence numbers for every route, so Y can notice that X's route to Z is an old one and is therefore to be discarded.

Time stamping

The sequence numbers are the most important feature of AODV for removing the old and invaluable information from the network. They works as a sort of timestamps and prevent the AODV protocol from the loop problem (see Appendix). The destination sequence number for each destination host is stored in the routing table, and is updated in the routing table when the host receives the message with a greater sequence number. The host can change its own destination sequence number if it offers a new route to itself, or if some route expires or breaks.

Each host keeps its own sequence number, which is changed in two cases: before the node sends RREQ message, its own sequence number is incremented and when the node responds to a RREQ message by sending a RREP-message, its own sequence number becomes the maximum of the current sequence number and the node's sequence number in the received RREQ message

The reason is that if the sequence number of already registered is greater than that in the packet, the existing route is not up-to-date. The sequence numbers are not changed by sending HELLO messages.

Route discovery

Route discovery process starts when a source node does not have routing information for a node to be communicated with. Route discovery is initiated by broadcasting a RREQ message. The route is established when a RREP message is received. A source node may receive multiple RREP messages with different routes. It then update its routing entries if and only if the RREP has a greater sequence number, i.e. fresh information.

Reverse path setup

While transmitting RREQ messages through the network each node notes the reverse path to the source. When the destination node is found the RREP message will travel along this path, so no more broadcasts will be needed. For this purpose, the node on receiving RREQ packet from a neighbor records the address of this neighbor.

Forward path setup

When a broadcast RREQ packet arrives at a node having a route to the destination, the reverse path will be used for sending a RREP message. While transmitting this RREP message the forward path is setting up. One can say that this forward path is reverse to the reverse path. As soon as the forward path is built the data transmission can be started. Data packets waiting to be transmitted are buffered locally and transmitted in a FIFO-queue when a route is set up. After a RREP was forwarded by a node, it can receive another RREP. This new RREP will be either discarded or forwarded, depending on its destination sequence number: if the new RREP has a greater destination sequence number, then the route should be updated, and RREP is forwarded, if the destination sequence numbers in old and new RREPs are the same, but the new RREP has a smaller hop count, this new RREP should be preferred and forwarded, and, otherwise all later arriving RREPs will be discarded.

Optimal TTL sequence

Expanding ring search strategies for AODV were recently extensively studied, and different schemes were proposed. In [8] a RREQ is initiated with a small TTL value, followed by RREQs with incremented TTL values until a certain threshold is reached. Then, if no route is found, a RREQ is flooded across the whole network.

The authors of [10] tried to find the optimal initial TTL value, TTL step, and the TTL threshold value. They found that the use of initial and step TTL values greater than 1 result in reducing overhead and delay time. They found also that initial as well as step values depend of the network topology, but the threshold value does not.

Furthermore, other strategies were proposed to make the route discovery more efficient, e.g. using the history of hop-distance to decide which initial TTL value should be chosen.

Link breakage

Because nodes can move link breakages can occurs. If a node does not receive a HELLO message from one of his neighbors for specific amount of time called HELLO interval, then the entry for that neighbor in the table will be set as invalid and the RERR message will be generated to inform other nodes of this link breakage RRER messages inform all sources using a link when a failure occurs.

Implementations of AODV

There are many AODV routing protocol implementations, including Mad-hoc, AODVUCSB, AODV-UU, Kernel-AODV, and AODV-UIUC [11]. Each implementation was developed and designed independently, but they all perform the same operations. The first publicly available implementation of AODV was Mad-hoc. The Mad-hoc implementation resides completely in user-space and uses the snooping strategy to determine AODV events. Unfortunately, it is known to have bugs that cause it to fail to perform properly. Mad-hoc is no longer actively researched.

The first release of AODV-UCSB (University of California, Santa-Barbara) used the kernel modification strategy. AODV-UU has the same design as AODV-UCSB. The main protocol logic resides in a user-space daemon; in addition, AODV-

UU (Uppsala Univerisity) includes Internet gateway support.

The AODV-UIUC implementation is similar to AODV-UCSB and AODV-UU except 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. This is efficient because forwarded packets are handled immediately and fewer packets traverse the kernel to user-space boundary. All of the implementations discussed use HELLO messages to determine local connectivity and detect link breaks. In addition, all implementations (except Mad-hoc) support the expanding ring search and local repair optimizations.

III. PROPETIES OF AODV

Merits of AODV

The AODV routing protocol does not need any central administrative system to control the routing process. Reactive protocols like AODV tend to reduce the control traffic messages overhead at the cost of increased latency in finding new routes.

AODV reacts relatively fast to the topological changes in the network and updates only the nodes affected by these changes.

The HELLO messages supporting the routes maintenance are range-limited, so they do not cause unnecessary overhead in the network.

The AODV routing protocol saves storage place as well as energy. The destination node replies only once to the first request and ignores the rest. The routing table maintains at most one entry per destination.

If a node has to choose between two routes, the up-to-date route with a greater destination sequence number is always chosen. If routing table entry is not used recently, the entry is expired. A not valid route is deleted: the error packets reach all nodes using a failed link on its route to any destination.

Drawbacks of AODV

It is possible that a valid route is expired. Determining of a reasonable expiry time is difficult, because the nodes are mobile, and sources' sending rates may differ widely and can change dynamically from node to node.

Moreover, AODV can gather only a very limited amount of routing information, route learning is limited only to the source of any routing packets being forwarded. This causes AODV to rely on a route discovery flood more often, which may carry significant network overhead. Uncontrolled flooding generates many redundant transmissions which may cause so-called broadcast storm problem.

The performance of the AODV protocol without any misbehaving nodes is poor in larger networks. The main difference between small and large networks is the average path length. A long path is more vulnerable to link breakages and requires high control overhead for its maintenance.

Furthermore, as a size of a network grows, various performance metrics begin decreasing because of increasing administrative work, so-called administrative load.

AODV is vulnerable to various kinds of attacks, because it based on the assumption that all nodes will cooperate. Without this cooperation no route can be established and no packet can be forwarded. There are two main types of uncooperative nodes: malicious and selfish. Malicious nodes are either faulty and cannot follow the protocol, or are intentionally malicious and try to attack the network. Selfishness is noncooperation in certain network operations, f.e. dropping of packets which may affect the performance, but can save the battery power.

Comparison between AODV and OLSR

As a proactive protocol, OLSR produces large control traffic overhead on the network. This overhead consumes bandwidth. AODV surpasses OLSR in terms of storage and memory overhead because maintaining of the routing tables for the whole network requires much more communication between the nodes as well as much more storage than by using the AODV protocol. Also routes never been used are maintained.

As a reactive protocol the AODV has an evident weakness: its latency. The route discovery process can take some time. This delay can be a crucial factor in a network. Moreover, a proactive part of AODV (route maintenance, HELLO messages) increases the control messages' volume and the transmission cost. It also damages the reactive property of the AODV.

The scalability is another problem of AODV protocol: with growth of the network the average path length increases, and so does the probability that a link becomes invalid. Therefore the AODV is suited only for small and medium size networks, the scalability limit is about 1000 nodes. Simulations of Perkins' group shown that at 1000 nodes AODV performs poorly, only 25% packets are delivered. The number of RREQ messages grows fast linear with nodes population, and at 1000 nodes most packets are control messages.

So the AODV protocol can be used in networks with limited resources: bandwidth, energy, computational power, but with a limited number of nodes, too. AODV is much more adaptable to highly dynamic topologies as OLSR does.

IV. CONCLUSION

In this paper the AODV routing protocol has been reviewed. As a reactive protocol AODV transmits network information only on-demand. The limited proactive part is the route maintenance (HELLO messages). The AODV protocol is loop-free and avoids the counting to infinity problem by the use of sequence numbers. This protocol offers quick adaptation to mobile networks with low processing and low bandwidth utilization.

The weaknesses of AODV include its latency and scalability.

The main conclusion of this paper is that the choice of which protocol to use depends on the properties of the network.

References

1. Schiller J. Mobile Communications. Addison Wesley, 2nd edition, 2003.
2. Royer E.M. Perkins C.E. Ad-hoc on-demand distance vector routing. Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, p.90, 1999.
3. Das S. Perkins C.E., Belding-Royer E.M. Ad-hoc on-demand distance vector (aodv) routing. RFC 3561, IETF Network Working Group, 2003.
4. AODV homepage. http://moment.cs.ucsb.edu/aodv/aodv.html.
5. Pucha H. Hu Y.C. Koutsonikolas D., Das S.M. On optimal ttl sequence-based route discovery in manets. volume vol.9, p.923, 2005.
6. Schneider S. Kaddoura M., Ramanujan R. Routing optimization techniques for wireless ad hoc networks. Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks SNPD/SAWN 2005, p.454, 2005.
7. Liu M. Chang N. Revisiting the ttl-based controlled flooding search: Optimality and randomization. In Proc. of ACM MobiCom, September, 2003.
8. Perkins C.E. Lee S.-J., Belding-Royer E.M. Scalability study of the ad-hoc on-demand distance vector routing protocol. Int. J. Netw. Manag., 13(2), 2003.
9. Tanenbaum A.S. Computer Networks. Prentice Hall International, 4th edition, 2003.
10. Jha S. Hassan J. On the optimization trade-offs of expanding ring search. Springer Verlag, 2004.
11. Belding-Royer E.M. Chakeres I.D. Aodv routing protocol implementation design. 2003.
12. There are also so-called hybrid protocols.

Appendix

Proof by contradiction: Let $\{K_1, \ldots, K_n\}$ be a loop in a route from any source node to any destination node. That means that these nodes are chained to each other. Assume without loss of generality that

$$\text{next Hop}(K_i) = K_{i+1}, \quad \text{for } 1 \le i \le m, \text{ where } K_{m+1} \equiv K_1 \qquad \textbf{(E.1)}$$

Then, from the definition of AODV destination sequence numbers, we have

$$\text{DestSeq No}(K_i) \le \text{DestSeq No}(K_{i+1}) \Rightarrow \text{DestSeq No}(K_i) = \text{DestSeq No}(K_j), \text{ for all } i,j \text{ belongs to } \{1,.., m\}$$

This means that information about the destination node was obtained from the same RREP message. Taking into account (E.1) and the definition of the hop count, we get

$$\text{hopCount}(K_i) = \text{hopCount}(K_{i+1}) + 1 \qquad \textbf{(E.2)}$$

Therefore,

$$\left.\begin{array}{l} \text{hopCount}(K_1) = \text{hopCount}(K_m) + m - 1 \\ \\ \text{hopCount}(K_m) = \text{hopCount}(K_1) + 1 \end{array}\right| \Rightarrow m = 0$$