MAD-HOC

Technical Documentation

Fredrik Lilieblad
Oskar Mattsson
Petra Nylund
Dan Ouchterlony
Anders Roxenhag

Abstract

An ad-hoc network is the cooperative engagement of a collection of mobile nodes without the required intervention of any centralized access point or existing infrastructure. There are several protocol drafts under evaluation. We have studied one of these, Ad-hoc On-demand Distance Vector routing protocol (AODV). It provides loop-free routes through the use of sequence numbers associated each route.

We propose an addition to the AODV draft version 5. The addition is that bidirectional routes will be set up when two-way communication is expected, which is almost always the case.

We have also made a prototype of an implementation of the protocol. It has been done in user space and relies wholly on system calls for its interaction with the kernel. Mobile wireless environments often have nodes of different strengths, resulting in asymmetric links. In the implementation there are some changes with regard to the AODV draft to cope with this problem. Sequence numbers are updated more often to avoid deadlocks. The Hello messages are changed from broadcasted Route Replies to broadcasted Route Requests. This will ensure that the created links are bidirectional.

Contents

1 Introduction

The developments in wireless communication devices have given rise to the possibility to form wireless networks between cooperating nodes. In the IETF community the MANET (Mobile Ad-hoc Networks) Working Group for routing have been working to produce a set of standards for routing protocols in ad-hoc networks. In the paper Routing Protocol Performance Issues and Evaluation Considerations by S. Corson and J. Macker they describe the essence of ad-hoc networking in these sentences:

> The vision of mobile ad-hoc networking is to support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multi hop topologies which are likely composed of relatively bandwidth-constrained wireless links.

Ad-hoc networking is still in its childhood and not many implementations exists on the market today. We will here present an implementation of one of the most promising ad-hoc routing protocols, the AODV protocol. Our implementation is to our knowledge the only open source implementation of this protocol that exist today and the only one that supports the AODV draft number 5 [1]. In addition to the implementation there are several suggested additions to this protocol which have proven successful under empirical testing by our test team.

First we will present a brief background to ad-hoc routing and the AODV protocol, then we will concentrate on the workings and specifics of our implementation. The suggested additions to the protocol will be presented together with our views on future development.

1.1 Background

Encyclopedia Britannica on ad-hoc:
[1]**ad-hoc**

adverb
Pronunciation: 'ad-'häk, -'hOk; 'äd-'hOk
Etymology: Latin, for this
Date: 1659
: for the particular end or case at hand without consideration of wider application

[2]**ad-hoc**

adjective
Date: 1879
1 a : concerned with a particular end or purpose <an ad-hoc investigating committee> **b :** formed or used for specific or immediate problems or needs <ad-hoc solutions>
2 : fashioned from whatever is immediately available : IMPROVISED <large ad-hoc parades and demonstrations – Nat Hentoff>

An ad-hoc network is a collection of cooperating mobile nodes in a wireless environment forming a temporary network without need for any infrastructure. To extend the range of wireless hosts ad-hoc routing protocols are used to enable multiple hops for the data sent by a node.

There are essentially two different strategies for routing. The first concept is pro active routing. In this case the node exchanges information with other nodes about routes, always keeping a static routing table. The second strategy is on demand routing. This implies that the node tries to discover a route to another node only when it is needed, i.e. no route exists until it is asked for. Through on demand routing the node does not have to wait for periodic updates (like in the Routing Information Protocol) from its neighbors to acquire a route.

1.2 Related studies

Thesis work at Uppsala University by Purser, Kevin Ericsson Radio systems, implementation of early draft of AODV.
Work done by at Carnegie Mellon University implementing the protocol DSR.

1.3 Scope

This paper will cover the technical aspects of our implementation of the AODV protocol. It will also cover our proposed additions to the workings of the protocol as well as suggest some future developments of the implementation.

2 The workings of AODV

AODV stands for Ad-hoc On-demand Distance Vector routing and is, as it states, an on demand protocol. It provides loop-free routes through the use of sequence numbers associated to each route. In short, if A needs a route to B it broadcasts a ROUTE REQUEST message. Each node that receives this message, and does not have a route to B, rebroadcasts it.

The node also keeps track of the number of hops the message has made, as well as remembering who sent it the broadcast.

If a node has the route to B it replies by unicasting a ROUTE REPLY back to the node it received the request from. The reply is then forwarded back to A by unicasting it to the next hop towards A. This establishes a uni-directional route(asymmetrical link). For a bi-directional route(symmetrical link) this procedure will need to be repeated in the reverse direction(See our additions 4.2).

To achieve faster convergence in the net, and thus higher mobility, a ROUTE ERROR message can be broadcasted on to the net in the case of a link breakage. Hosts that receive the error message remove the route and re-broadcasts the error messages to all nodes with information added about new unreachable destinations.

3 Overview of the implementation

The aodv daemon is a user space process written for the Linux kernel. The aodv daemon relies wholly on system calls for its interaction with the kernel. It relies only on the functionality provided by the kernel such as the IP stack and its IP routing. This implementation does not deal with the multicasting part of the AODV protocol.

The implementation consists of two processes: the **aodv daemon** and the **packet capture process**. The aodv daemon is a top loop, controlling the actions to be taken and what modules to be called.

3.1 The modules

Modules used by the aodv daemon are (as seen in the picture):

- **aodv daemon**
- **receive/generate route request**
- **receive/generate route reply**
- **receive/generate route error**
- **aodv routing table**
- **packet resend queue**
- **packet capture process**

For the workings of the protocol see A.2 to A.7.

3.1.1 General information

Since much in the system relies on time, the time is measured as epoch time in milliseconds. The time in the system is represented as a 64 bit unsigned integer. Most of the communication between the different modules are done by passing of pointers to structs. The whole system is designed so that there should be no deadlocks or read/modify/write problem. The daemon is a sequential program. The only other process, the packet capture process, communicates with the aodv daemon via messages through uni-directional message passing. The two processes do not share any data.

3.1.2 The aodv daemon

This process works as a dispatcher for the incoming packets. It listens(and sends) on port 1303[1](`AODV_PORT`) on a specified interface for incoming packets, it then uses the first 8 bits in the packet to determine which type of packet it has received. (Either a RREQ, a RREP or a RERR as defined in the aodv-draft [1].) It then passes the AODV packet together with additional IP header information to the module which handles the appropriate type of packet. The aodv daemon also listens to the packet resend timer and the packet capture process. The resend timer alerts the aodv daemon when a sent packet should be resent (i.e. a timer has expired) in which the aodv daemon takes the appropriate action, which is to resend the packet. The aodv daemon also listens to the packet capture process which reports packets forwarded by the kernel, received ICMP packets and sent ARP requests.

The aodv daemon must not restart immediately after a crash. Instead it enters reboot mode, which means that all AODV packets are ignored, and that all data packets are answered with a RERR. The reboot mode has to last at least `DELETE_PERIOD` number of milliseconds. This is controlled by having two files in `/var/lock`: the crash file(`aodv_lock`), and the time guard file(`aodv_time`). The crash file is kept to ensure that the daemon hasn't crashed, and the time guard file is checked to prevent rapid exiting and restarting of the daemon.

3.1.3 Receive Route Request

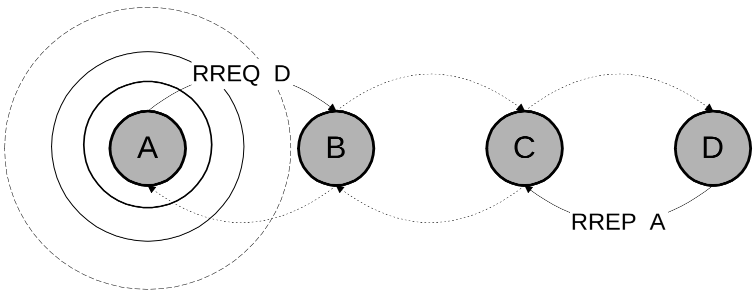
This module is called by the aodv daemon when a ROUTE REQUEST message is received on the active interface. The Receive Route Request module then determines if the request was for the node itself, or if it has a route to the requested host in its routing table, or if the request was for another node which it does not have a valid route for. If the ROUTE REQUEST was for the node itself, or the node has an active route to the requested destination, the node generates a ROUTE REPLY message with

[1] As in 2g1303.

the module Generate Route Reply. If the node does not have the route it simply re-broadcasts the message with an increased hop count and a decreased TTL. See appendix A.2 for the flowcharts of this module.

3.1.4 Generate Route Request

This module is called when the aodv daemon gets a message from the packet capture process indicating that the node has sent an ARP request for a destination. The ARP is used because it is the first indication that is observable form the outside that a node is looking for a route. This module is also used for sending HELLO messages (This is one of our additions to the workings of the protocol. See subsection 4.4). The use of HELLO messages are for detecting link breakage with neighbors. The sent Route Request is put in a buffer (packet resend queue) with a timer. If the timer expires the packet is resent up to a maximum of a given number of retries. See appendix A.3 for the flowcharts of this module.



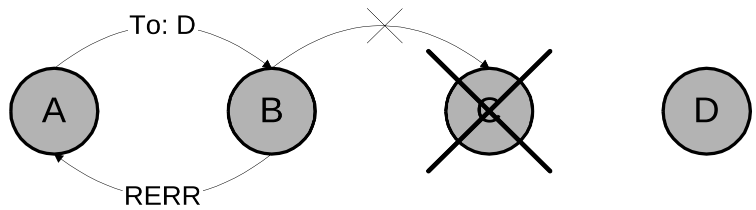
3.1.5 Receive Route Reply

This module is called when the aodv daemon receives a ROUTE REPLY message on the active interface. It checks to see if the acquired route is a better route (better is regarded as a route that has fewer hops or the same number of hops but a more recent route) than the one it already has (or doesn't have). If the route is a better one it updates its routing table. If the ROUTE REPLY was for the node itself, it first dequeues any ROUTE REQUEST for the destination that is in the resend queue. It then discards the message. If the ROUTE REPLY was not for the node itself it increases the hop count and decreases the TLL. It then sends(unicast) the messages to the next hop towards the destination of the ROUTE REPLY message. See appendix A.4 for the flowcharts of this module.

3.1.6 Generate Route Reply

This module is called from the Receive Route Request module when the host is going to reply to a request for a route. The module generates a ROUTE

REPLY message to be sent(unicast) to the address given by the Receive Route Request module. If the node is replying for a route to another node it then generates a ROUTE REPLY message to be sent to the node that was requested (This is one of our additions to the workings of the protocol. See subsection 4.2). If it is replying to a request for itself it also increases its source sequence number. See subsection 4.2 for en explanation of this behavior. See appendix A.5 for the flowcharts of this module.



3.1.7 Receive Route Error

This module is called when the aodv daemon receives a ROUTE ERROR message on the active interface. It then processes the unreachable destinations in the message one by one. For each unreachable destination in the ROUTE ERROR message it checks if the node it received the route error from is the next hop towards the unreachable destination. If so, it updates the sequence number to be the one in the ROUTE ERROR package and invalidates the route. If the node itself has precursors for the node that has been declared invalid it adds the destinations that are unreachable to the ROUTE ERROR messages. When all destinations in the ROUTE ERROR message has been processed it checks if it has added any destinations to the list in the message and if so it broadcasts the message on the active interface. See appendix A.6 for the flowcharts of this module.

3.1.8 Generate Route Error

This is divided into two different cases, the link breakage and the host unreachable case. The link breakage is called from the AODV Routing Table module when a neighboring node has not responded to HELLO messages during a period of time(`ALLOWED_HELLO_LOSS*HELLO_INTERVAL`). Then a ROUTE ERROR is broadcasted for this node. The host unreachable is called from the aodv daemon when it receives a message from the packet capture process, indicating that it has received a ICMP HOST UNREACHABLE for a node that is not the node itself and a ROUTE REQUEST to the node is not in the resend queue. During the reboot phase ROUTE ERRORS are sent for the node itself when it overhears ROUTE REQUESTS for itself. This is to speed up the invalidation of the node in the net after a reboot.

See appendix A.7 for the flowcharts of this module.

3.1.9 AODV Routing Table

This is the internal routing table for the AODV protocol. This is built as a linked list for easy implementation and debugging. When new routes become available through ROUTE REPLY messages they are entered into this table. It also handles the communication with the kernel's IP routing table. When a new route is inserted into the table it also checks to see if any other routes has become invalid or should be deleted. It contains, for each route in the table including the node itself, a tuple of:
`<dst_ip,dst_seq,broadcast_id,hop_cnt,`
`lst_hop_cnt,nxt_hop,precursors,lifetime,rt_flags>`

3.1.10 Packet Capture Process

The objective of this process is to capture IP and ARP packets and classify them for processing. The packet capture process is run parallel to the aodv daemon. It listens promiscuously on the active interface. It captures raw Ethernet frames which it then decodes into IP or ARP packets. No other type is regarded. The IP packets are divided into ICMP and other packets. The ICMP packets that are processed are type `ICMP_DEST_UNREACH` and code `ICMP_UNREACH_HOST` or `ICMP_UNREACH_HOST_UNKNOWN`. All other ICMP are regarded as ordinary IP packets. In ARP packets the process only deals with ARP requests from IP to Ethernet. When it has matched a packet it passes information of it to the aodv daemon by writing the message to a pipe which the aodv daemon reads from.

3.1.11 Packet Resend Queue

The packet resend queue is a timer driven queue. Packets are inserted into this queue with a timer value, that indicates when the packet should be resent. The timer runs in real time and uses the ALARM signal to interrupt the aodv daemon. Instead of directly dealing with the packet in the interrupt it writes a message to a pipe which is read by the aodv daemon and processed when there is time. The queue items are stored with an id tag for identification and a flag which tells what type of packet is stored at the position.

4 Our additions to AODV

In our efforts to implement AODV as a user space routing daemon we have come up with some useful additions to the workings of the protocol.

4.1 Sequence numbers

As the protocol is designed for symmetrical communication links some deadlocks can occur when using it in an environment with asymmetrical links. The situation is as follows: A and B are communicating with each other. There is a break on the line between A and B such that only B notices the break. B then invalidates the route entry for A in its routing table and increases its sequence number for A. Now A has sequence number n for itself and B has sequence number n+1 for A. Through the HELLO messages that A and B exchange A will never receive the update of its sequence number because through the HELLO message B will never ask for the route to A. This also means that when B has higher sequence number for A it will not listen to any messages sent by A (with the lower sequence number) because it will regard them as outdated.

The solution proposed here is to update the source sequence number more often, so that the node always has a higher sequence number than any other node for itself. This is accomplished by incrementing the node's sequence-number every time it announces itself onto the net, i.e. every sent ROUTE REPLY answering a request for the node itself. This leads to a more rapid increment of the source sequence number. This could pose a problem in the future when it is not defined in the AODV draft [1] what to do when the source sequence number reaches its maximum value $2^3 2$.

By doing this the node will always increase its source sequence number. Thereby it will not be deadlocked by another node with a higher destination sequence number for the node. The node will in the end always have the highest sequence number for itself.

4.2 Route Replies for other nodes

When dealing with AODV it only takes uni-directional connections into account when setting up routes to a node. The situation is as follows: if A wants to contact E but on the way to E there are the nodes B,C and D. B has a valid route to E and the fastest way to reach E is A→B→C→D→E. B then answers A's request for a route to E. E will not have a reverse route to A. If A only sends datagrams to E which do not produce any data in return from E to A, this is the optimal solution. In most cases, however, there will be an exchange of data between to nodes, even if it's only ACK:s. When E wants to contact A it has to do a ROUTE REQUEST for A, which is broadcasted through the net until it reaches B (because no-one on has the route to A, except A's neighbor). This broadcast-back behavior can be reduced by adding a simple mechanism to the protocol: When a node answers a ROUTE REQUEST on behalf of another node it should unicast a ROUTE REPLY to the destination node(E) with the source as ROUTE REPLY destination(E) and the ROUTE REPLY source as the querying node(A). This creates a bi-directional connection between A and E, significantly reducing the number of required broadcasts.

4.3 The bi/uni-directional flag for ROUTE REQUEST messages

To the solution of the problem presented in section 4.2 an addition can be made to control when a replying host should send a ROUTE REPLY to the destination of the ROUTE REQUEST. A flag could be set in the ROUTE REQUEST message header indicating that the route requested should be a bi-directional one. When a node replies to a ROUTE REQUEST it should check to see if the sender has requested a bi-directional link. In this case it should follow the behavior given in section 4.2. In the header of the ROUTE REQUEST there exists a reserved field of 8 bits. One of these 8 bits could easily be used for this purpose. This would not affect any other parts of the protocol.

By request from the author of [1] an addition to the draft has been submitted. See appendix A.1.

4.4 The HELLO messages

The proposed way for a node to keep track of it's neighbors is for it to send HELLO messages at intervals. The HELLO messages in the AODV draft [1] is a ROUTE REPLY broadcasted by a node to all nodes with the node itself as the destination node and the nodes own sequence number. One can here clearly see that the link which the HELLO message set up to it's neighbors is only uni-directional. If a node hears a HELLO message sent by another node it does not know if it really can send data to it (the case of a strong and a weak sender).

In the implementation this is replaced by a two way HELLO message. The ROUTE REPLY is replaced by a ROUTE REQUEST message. The ROUTE REQUEST message has the destination IP address 255.255.255.255 and the source address as it's own IP address and the TTL set to one. When a node receives a ROUTE REQUEST message asking for 255.255.255.255 it will respond as if it was asking for the node in question. This will ensure a two directional HELLO message resulting in a bi-directional link being established between the neighbors. The resulting overhead of this procedure will be that for each broadcasted HELLO messages one unicasted message will be sent back by every node.

5 Specifics of implementation

Since this is a user space implementation of AODV there are some aspects that have to be taken into account.

5.1 Lookups in the routing table

How does one detect (in user space) when an application is trying to look up a route (in kernel space) that does not exist in the routing table? This proved to be quite difficult without changing the kernel code. In previous versions of the Linux kernel the routing message `RTM_MISS` was sent to user space processes when a lookup fails in the routing table. In more recent versions of Linux this code has been re-written and the routing messages replaced with routing multicast groups in which none includes the information of failed route lookups. The solution to this in the implementation is to look for sent ARP requests on the interface that is running the aodv daemon. When an ARP request is sent the aodv daemon sends a ROUTE REQUEST for the requested address. The address is then added to the routing table when a ROUTE REPLY for the address is returned.

5.2 Buffering of packets

While waiting for the ROUTE REPLY to arrive, in the response to the ROUTE REQUEST send by the detection of the ARP request, the implementation does not buffer packets. This will result in the loss of the packets sent during the period between the ROUTE REQUEST and the ROUTE REPLY. To be able to buffer packets that do not have a route requires modifications to be made in the kernel.

This approach works because the layers residing above the IP layer do not rely on the performance of the lower layers. The higher layer protocols often use resend schemes to correct for the imperfection of the lower layer. Thereby they overcome the loss of the first packets that the aodv daemon could not handle.

5.3 Detection of unreachable hosts

The detection of unreachable hosts, who are not neighbors, is determined by listening to ICMP packets on the interface declaring `UNREACHABLE_DEST`.

6 Building, Installing and Running

6.1 Requirements

Root privileges will be needed to make the changes that have to be done in order to run the daemon. Even the daemon will require root privileges to run since it uses raw sockets and promiscuous mode on the interface.

6.1.1 Software

Linux 2.2 kernel (not tested on any other) with ip_forwarding enabled. Might as well function on any other Linux kernel.
Libcap 0.4.

6.1.2 Hardware

Ethernet interface such as 10BaseT or WaveLan.

6.2 Building

Builds using the gcc compiler, it will not build with cc when GNU byte addressing is used in some of the header files.
To build the aodv_daemon run make.
This will produce the file aodv_daemon.
To enable logging and debugging info, edit the Makefile to compile with the flags '-DMSGLOG' or '-DDEBUG'. The MSGLOG flag makes the program save a file in the working directory called 'msglog.txt'.

6.3 Running

Make sure you have root privileges before you set up your system.
To run the aodv_daemon program for ad-hoc networking set the desired interface in ad-hoc mode (if its a WaveLAN card). For pcmcia cards the file to change is `/etc/pcmcia/config.opts`. Change or add a line that says `module "wvlan_cs" opts "port_type=3 channel=1 station_name=MY_PC"`. Make sure you don't have any other lines configuring the wvlan_cs opts such as network_name. Now when you restart your WaveLan card it is in ad-hoc mode.
Assign an ip to the card 'ifconfig wvlan0 172.16.0.1 up' for example.
Add the broadcast address to the interface with
`route add 255.255.255.255 wvlan0`.
Turn on ip forwarding in the kernel with
`echo 1 > /proc/sys/net/ipv4/ip_forward`.
Run the aodv daemon with `aodv_daemon wvlan0` or appropriate interface.

When in operation the aodv daemon has a minimalistic user interface. Hitting enter while running will present a small menu with some commands that can be given to the aodv daemon. The menu looks like this:

```
gen_rreq:xxx.xxx.xxx.xxx
print_rt
add_rt:dst_ip:dst_seq:broadcast_id:hop_cnt:lst_hop_cnt:nxt_hop:lifetime:rt_flags
link_break:xxx.xxx.xxx.xxxn
Command:
```

The commands are executed by pressing enter. The execution is halted during the display of the menu. Commands can be given without having displayed the menu. The `xxx.xxx.xxx.xxx` are replaced by an ip number. The arguments to `add_rt` are the same tuple as in the routing table. `rt_flags` are always 0.

7 About the source code

The source code is compliant with the standard set by [2].

7.1 Future developments

Due to the limited time that was to our disposal during the development of the daemon there are still some things that can be made better or added to the code.

The aodv routing table is at the present a linked list this should be made into a more efficient data structure. Some kind of a hash table or binary tree, like Radix, is preferred.

There is no way at the present to see that routing table lookup has failed. Looking for ARP requests is not the best way, there may be some other way to do this.

7.2 Known bugs

At the moment there are no known bugs.
Please report bugs to mad-hoc-bugs@flyinglinux.net.
Bug reports and updates will be available from mad-hoc.flyinglinux.net.

7.3 License

This software is licensed under the GNU General Public License also known as GPL. The license text is included in the appendix, see A.8.

8 Future developments

For future work with this implementation these points could be a lead.
Add support for :

- Routing decision policies.
- Security.
- Multiple interfaces.
- Multicasting.

- Gateways.
- Buffering of ip packets.
- Link level feedback.

Move parts of the implementation into the kernel. This could be made trough creating a virtual interface for the aodv protocol. For this interface most of the existing IP stack could be reused. Then there would only be copying of data between the virtual interface and a real interface.
Make a whole kernel implementation.

9 References

References

[1] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das:
Ad-Hoc On-Demand Distance Vector (AODV) Routing, 10 March 2000, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-05.txt

[2] Brian W. Kernighan, Dennis M. Ritchie:
The C Programming Language, 1998 Prentice Hall International.

10 Conclusion

The implementation presented in this paper has the potential to be a base for testing of aodv and other on demand routing protocols. It is scalably and modularly built with extensive documentation and would be easy for anyone to adopt and make modifications to. The additions to AODV proposed in this paper will hopefully be integrated into the future drafts and standard of AODV. Especially since a draft formulation of our propositions has been requested by Charles E. Perkins, creator of the AODV protocol.

11 Glossary

AODV Ad-Hoc On-demand Distance Vector.

epoch time Time used in UNIX type system. It's value is the number of seconds elapsed since.

multiple hops The traversing of a packet over several forwarding nodes.

loop-free That the protocol ensures that no circular routes exist.

daemon A user space process providing some service in the background.

route request An AODV messages for requesting a route to a node.

route reply An AODV messages for answering a route request.

route error An AODV messages when announcing that routes have broken.

packet capture A program that looks at all the passing packets on an interface.

ICMP Internet Control Message Protocol.

ARP Address Resolution Protocol.

node A unit in the network (a host).

asymmetrical link A link that only can carry information in one direction.

sequence number A number used in AODV to indicate the most recent route to a node.

ACK Acknowledgement.

neighbor A node that can be reached without using any intermediate nodes.

bi-directional A route that passes information in both directions.

user space Not kernel space.

promiscuously Listens to all packets on the interface, not only packets destined for the address assigned to the interface.

frag The killing of another player in a first person shooter.

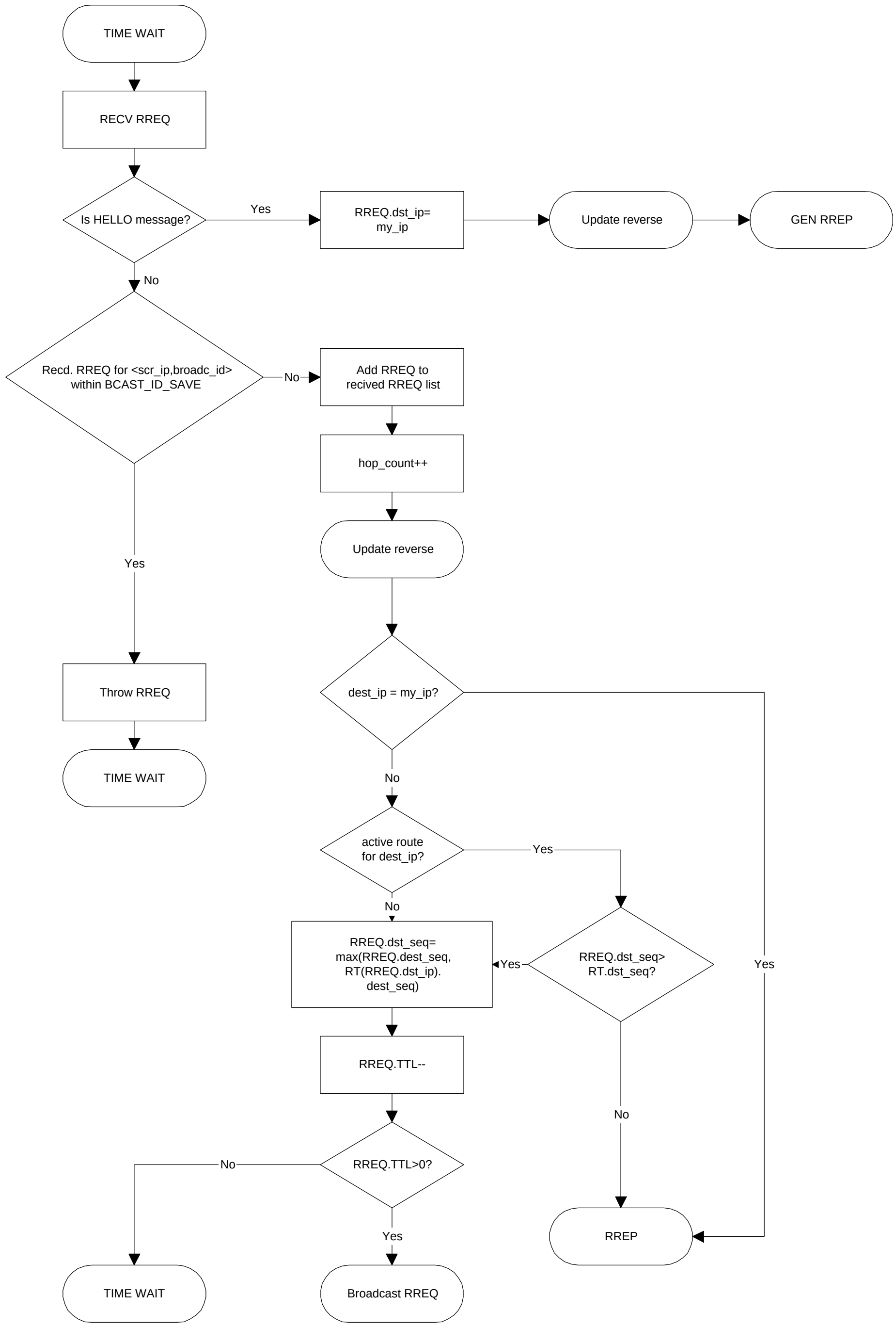
A Appendix

A.1 Submitted addition to the AODV draft

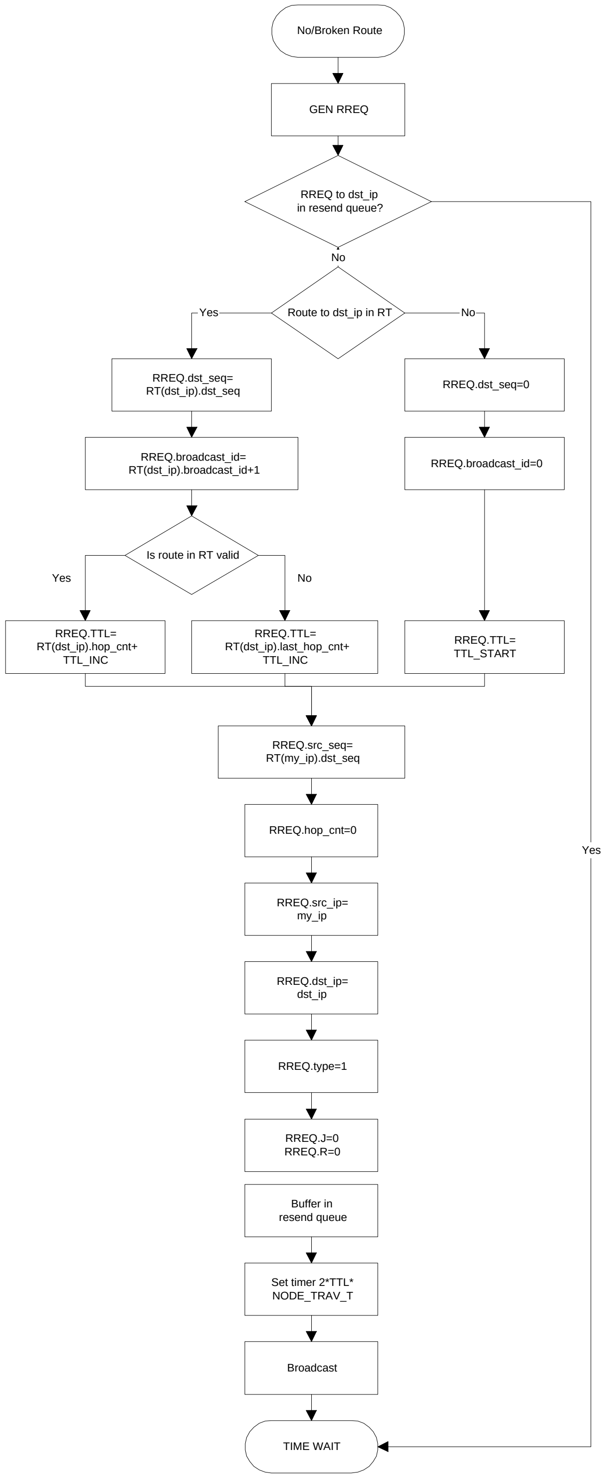
```
-- page 5 --
4.

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |J|R|B|      Reserved           |   Hop Count   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Broadcast ID                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination IP Address                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Destination Sequence Number                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source IP Address                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Source Sequence Number                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

-- page 6 --
4.

      B      Bidirectional flag; set when source node
     requests a route intended for two-way
     communication.

-- page 12 --
9.2

   The bidirectional flag SHOULD be set if a RREQ message for a two-way
   connection is sent. Typically the B flag is set for requests
   originating from TCP packets, and is not set for UDP packets.

-- page 15 --
9.4

   In addition, the generating node puts the last hop node (from which it
   received the RREQ, as indicated by the source IP address field in the
   IP header) into the precursor list for the forward path route entry.
   (This is the entry for the Destination IP Address).

   If the generating node is not the destination node, and the B
```

(bidirectional) flag in the received RREQ is set, the generating node should send an additional RREP message. The target of this second RREP message is the node indicated by the destination IP address in the RREQ message. The generating node places its distance in hops from the source of the RREQ (indicated by the source IP address field in the RREQ message) in the Hop Count field of the RREP message. The Destination IP Address and the Destination Sequence Number of the second RREP are copied from the Source IP Address and Source Sequence Number of the RREQ. The Source IP Address of the second RREP is copied from the Destination IP Address of the RREQ. The lifetime is set to ACTIVE_ROUTE_TIMEOUT. It also puts the next hop towards the destination in the precursor list for the reverse route entry. (This is the entry for Source IP Address.) The generating node consults its route table entry for the destination node of the second RREP message to determine its next hop, and then forwards the second RREP towards the destination.

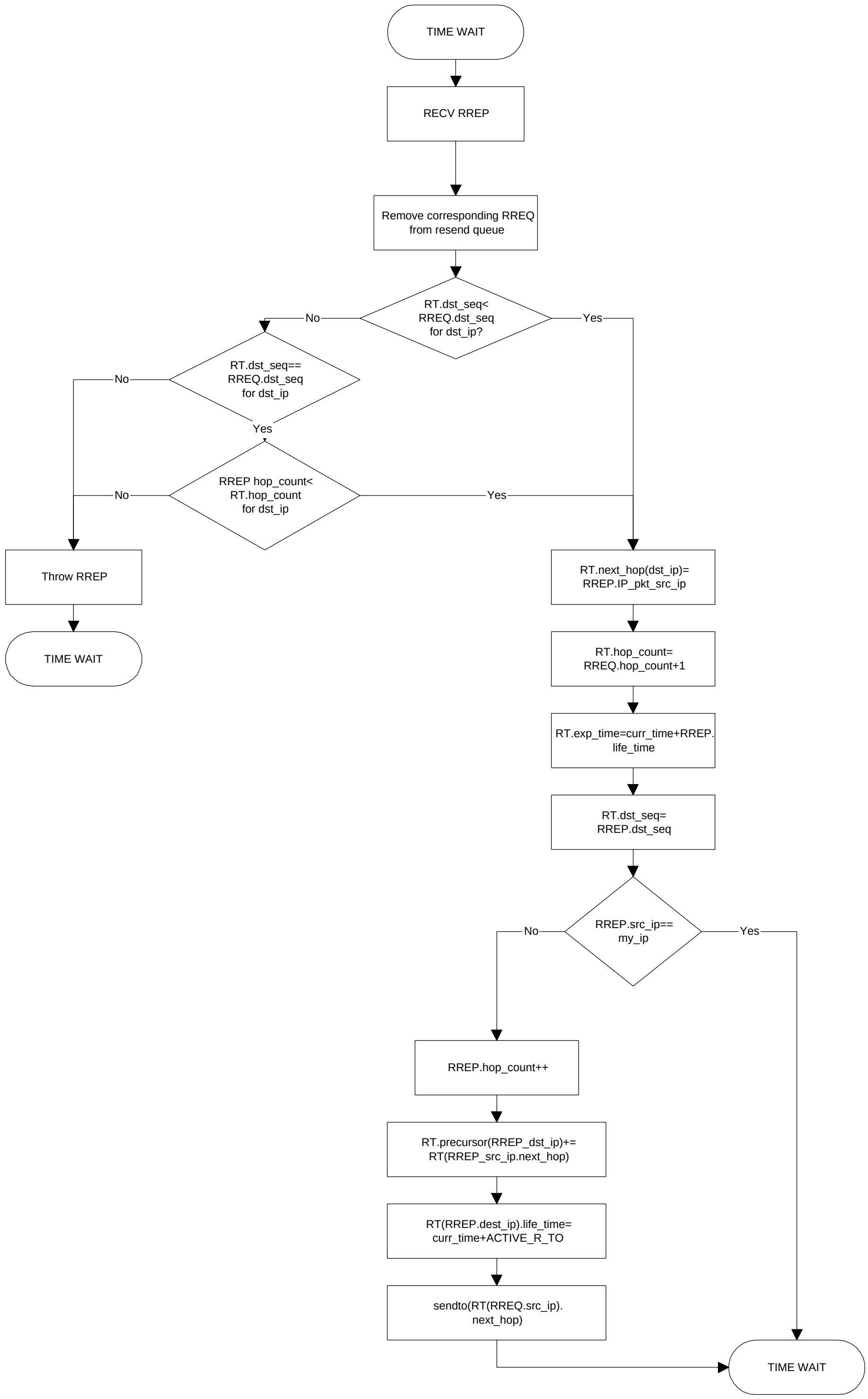
A.2 Receive Route Request



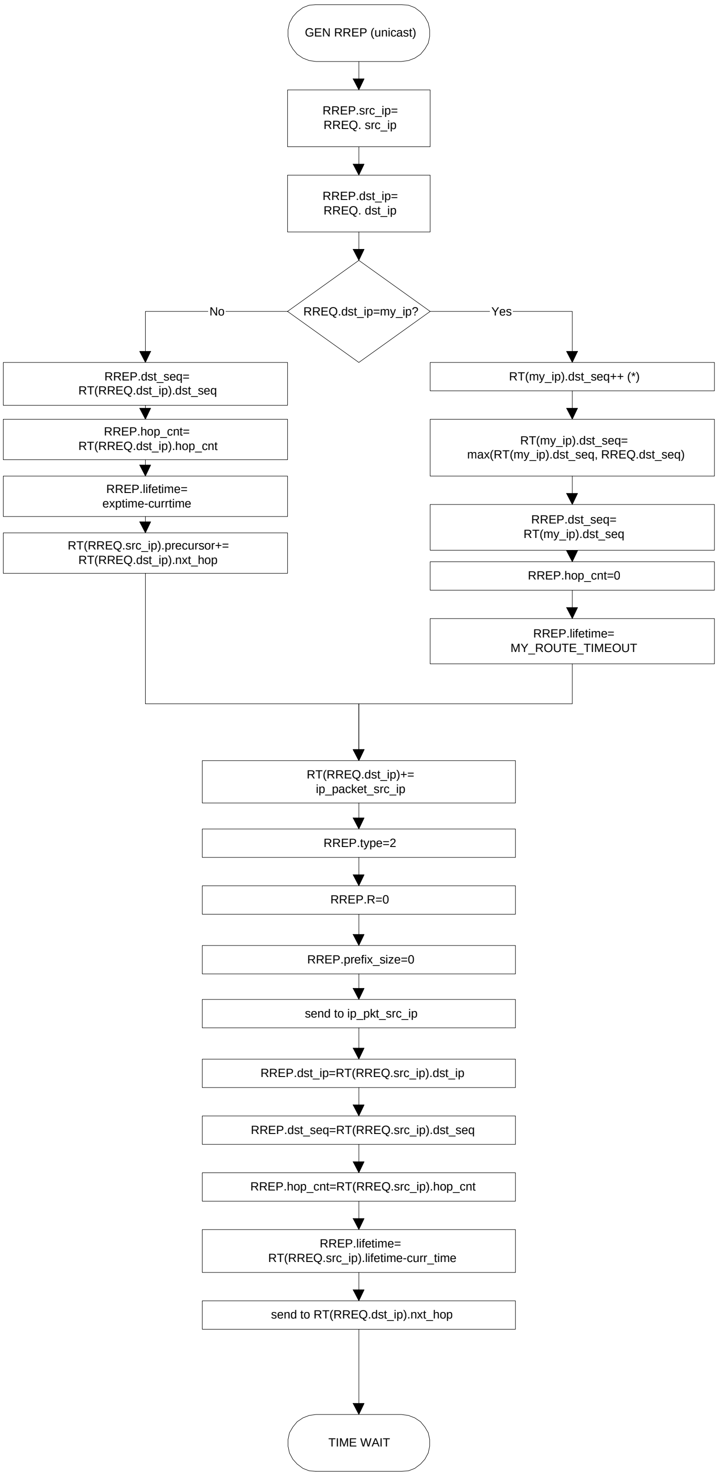
A.3 Generate Route Request



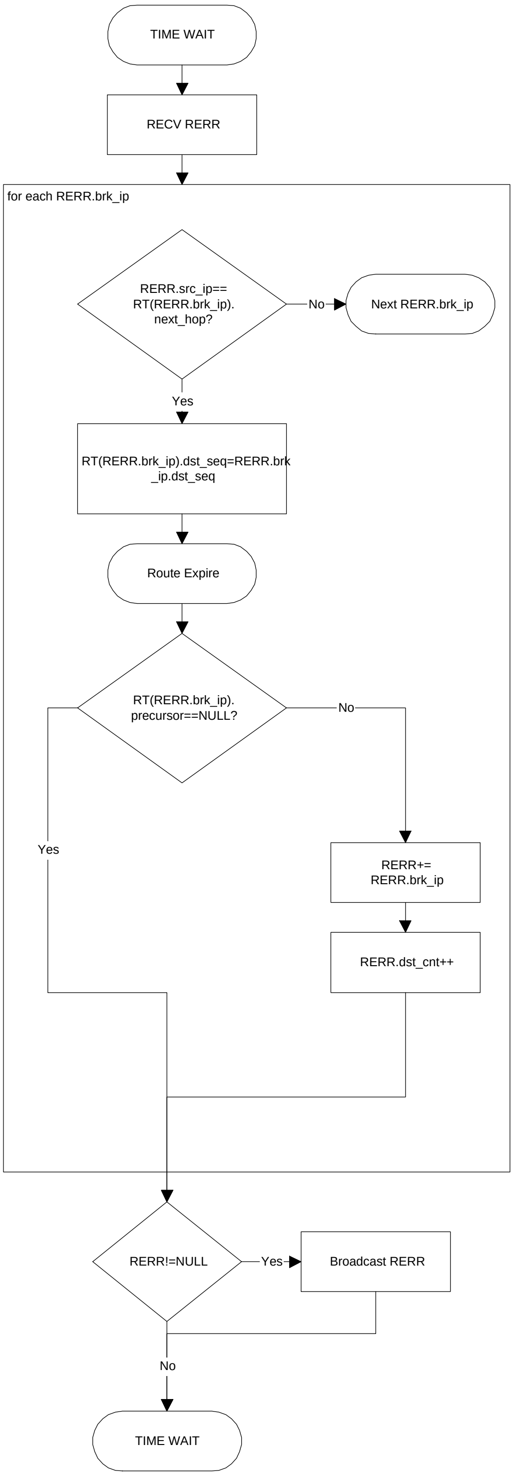
A.4 Receive Route Reply



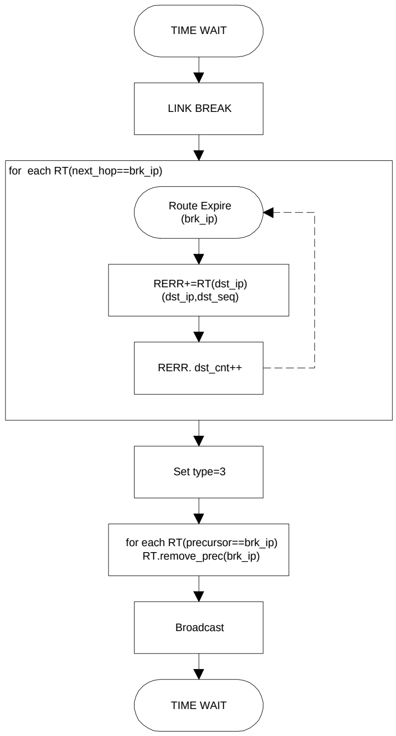
A.5 Generate Route Reply

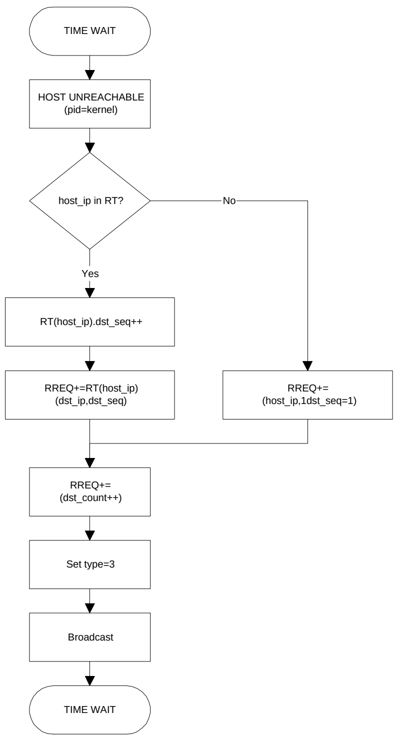


A.6 Receive Route Error



A.7 Generate Route Error





A.8 The GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may

```
be called something other than ‘show w’ and ‘show c’; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```