OLSR Simulation, Implementation and Ad Hoc Sensor Network Application

Christopher Dearlove (chris.dearlove@baesystems.com) BAE SYSTEMS Advanced Technology Centre, UK.

Abstract

A summary is given of the key decisions in the creation of a complete implementation of OLSR (RFC 3626). This implementation is suitable for simulation, in particular in OPNET, and for real time use, in particular under Linux. An example simulation result, illustrating a proprietary extension to the protocol is given, and also a description of the demonstration of an ad hoc sensor network. Finally additional requirements for certain ad hoc networks are included.

1. Introduction

Optimised Link State Routing (OLSR) is a routing protocol for a mobile ad hoc network (MANET). OLSR is defined by the experimental standard RFC 3626 [1]. This paper describes an implementation of OLSR and its use both for simulation and as a real time implementation of the routing protocol, for which it is used in an ad hoc sensor network demonstration.

2. History

Prior to RFC 3626, OLSR was defined by a sequence of Internet Drafts published by the Manet group of the Internet Engineering Task Force (IETF). The software described here started as an implementation of version 5 of OLSR at a time when no public implementation of this version of OLSR was available. This implementation was later updated to version 7 and then version 11 (equivalent to RFC 3626) of OLSR. Some observations based on creating and using this implementation were fed back to the OLSR authors.

3. Software Requirements

This implementation of OLSR started with the following requirements

- A definition of a framework for a generic ad hoc routing protocol (AHRP).
- An implementation of OLSR within this framework.
- The ability to simulate any such AHRP, in particular using the simulation tool OPNET [2].
- The ability to implement any such AHRP in real time, in particular using an IEEE 802.11b equipped laptop (or, later, PDA) using Linux.
- To include IPv4 and IPv6 options.
- To be able to interwork with other implementations of OLSR.

The selected framework for a generic AHRP assumed a protocol running over UDP. The framework was to include reactive protocols such as AODV [3] and other proactive protocols such as TBRPF [4]. It did not allow for protocols such as DSR [5] which modify the IP layer. Currently only OLSR has been implemented using this framework.

The real time operation referred to is "soft" real time. The code was written with reasonable regard for efficiency, but no detailed optimisations, and observed, for networks thus far implemented, to be more than fast enough.

Later these requirements were augmented with the following additions

- To include some compliant proprietary extensions to OLSR.
- To support modified and extended versions of OLSR.
- To include dynamic variation of OLSR parameters.

The first of these additional requirements is discussed further below. The second of these additional requirements includes modifications to OLSR such as overriding the usual MPR selection algorithm, or extensions such as Secure OLSR [6].

4. Software Design

Given the requirement for a generic framework, and that OPNET directly supports only C and C++ code, the obvious design of the software was in C++, with the AHRP framework being represented by an abstract base class `Ahrp` and OLSR implemented as a class `Olsr`. Class relationships are shown in Figure 1 below; the two greyed out classes are hypothetical.

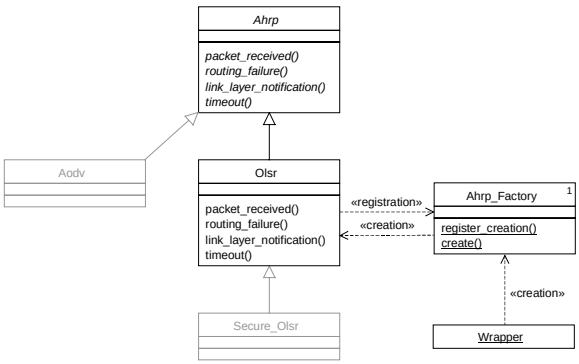

Figure 1: AHRP and OLSR Classes and Creation

The interface shown above for classes `Ahrp` and `Olsr` includes the most important class member functions, but is incomplete: it excludes parameter variation as well as features such as dynamic addition and removal of gateways. The interface includes a function, `routing failure()`, required by a reactive protocol (to initiate route finding) whose implementation in a proactive protocol is trivial. The interface by which a modified version of OLSR may be created is also not shown or discussed here.

Figure 1 also includes a "wrapper" object. This is the code specific to the platform on which the OLSR, or other routing protocol, is to be used. For OPNET this is a process model, for Linux this is code running in user space. The wrapper object is completely independent of the selected routing protocol; this is

accomplished using usual runtime polymorphism as shown in Figure 2 below. For routing protocol creation this uses an object factory, of the particular style described in [7] shown in Figure 1.

The wrapper includes, in particular, writing to the IP routing table and handling outgoing packets (passing them to UDP). As shown in Figure 2 below for Linux (OPNET is similar) these are managed by "auxiliary" classes which are specific to the wrapper, but with abstract base classes used by the routing protocol.

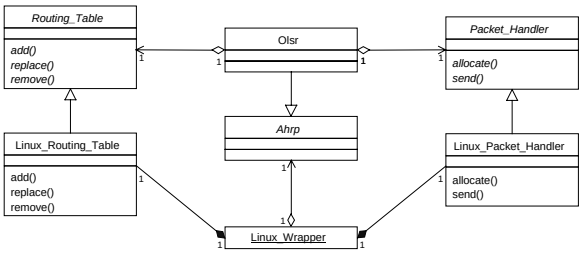

Figure 2: AHRP and OLSR Class Usage

Provision for IPv4 and IPv6 is, unlike all other options, handled by a compilation switch.

5. OLSR Compliance

The OLSR software has been designed to be fully compliant with RFC 3626, although at the time of writing this has not been verified. It includes all the options included in RFC 3626, in particular

- It supports multiple interfaces.
- It supports external gateways (host and network associations). It includes an option for dynamically adding and removing these (constrained by limitations of OLSR with regard to the latter).
- It includes link quality, defined either by link layer notification or by tracking received and lost packets and HELLO messages.
- All parameters, not just willingness, are fully dynamically variable.
- It supports IPv4 and IPv6 by separate compilation.

The software does however currently have the following limitations

- It does not piggyback outgoing messages into a single packet. However it does properly process incoming packets containing piggybacked messages.
- There is no message or packet size control and messages are not fragmented (although, with differing values of the OLSR parameters `HELLO_INTERVAL` and `REFRESH_INTERVAL`, HELLO messages may be incomplete). This does avoid certain problems that can arise with fragmented TC messages. However it does properly process incoming fragmented messages.

6. OLSR Extensions

In addition to extending the OLSR definition in areas noted above (dynamic gateways and parameters) the software implements the following optional proprietary, but compliant, extensions to OLSR. (In the latter two cases the extension may not strictly be compliant, but should seamlessly interwork with fully compliant implementations.)

- Optional minimum message intervals discussed further below.
- Options to allow reuse of the existing MPR Set if possible, despite possible recalculation (including options to reuse even if a strictly better set can be found).
- Link layer notification options to mix link layer notification and message tracking. (This is, in particular, to allow link quality to decline when link layer notifications cease due to a link breaking.)
- Accelerated protocol set updates (of Interface Association Set by HELLO messages and of Two Hop Neighbour Set by MID messages).

7. Simulation

This paper does not include extensive simulation results; some simulation results produced using this and other software are reported in [8]. As an example thereof, and to illustrate the minimum interval option noted above, a single result is included here. The simulation is of a number of identical nodes (10, 30 or 50) in a fixed size square world moving using a random walk with reflection. For details see [8].

The minimum intervals used provide a minimum separation between messages of the same type. Thus for example if a neighbourhood change is observed such that a new HELLO message should be sent and if the last such message was sent less than the HELLO message minimum interval ago, then sending is postponed until the minimum interval after the previous message (modified by jitter as appropriate). Thus a minimum interval of zero causes messages to be sent whenever a significant change is observed, whereas a minimum interval equal to the usual message interval means that only regularly scheduled messages are sent. A minimum interval between these values provides a soft transition between these two behaviours.

Separate minimum intervals are supported for each message type, but the results in Figure 3 below scale all minimum intervals together. The results for minimum intervals show an insignificant reduction in performance (proportion of data packets received) but a great reduction in overhead in this particular, high mobility, case.

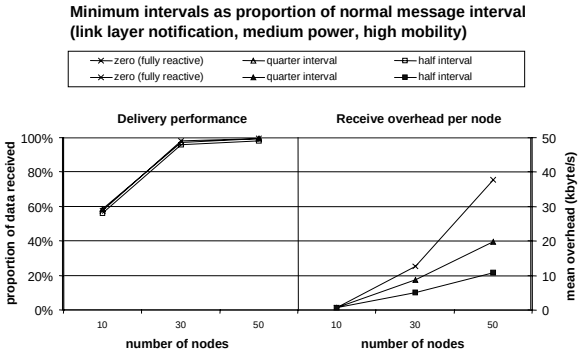

Figure 3: Simulation Results (Minimum Intervals)

It may be noted that these results show poor performance for the 10 node network. This is due to the fixed size of the world and hence reduced node density for fewer nodes. Results in [8] show how this is exacerbated by use of link layer notification, as here, which improves performance with more nodes (greater density) but may cause network fragmentation with fewer nodes (lower density). Some form of dynamic control over use of link layer notification might be the solution to this problem.

8. Sensor Network Applications

A particular case of interest is an ad hoc network of nodes, some or all of which contain sensors. These may just be stationary ground sensor networks, or could include mobile nodes, up to and including unmanned aerial vehicles (UAVs). Even while the network is stationary, there is still a place for a mobile ad hoc network protocol

- The network is likely to be unplanned, and deployed in an ad hoc manner, possibly for example being scattered from the air. This requires ad hoc configuration of the network.
- The network may be subject to loss, possibly due to battery exhaustion or damage (accidental or deliberate).
- Radio transmission conditions may necessitate a change in network topology. This may be deliberate, due to jamming, or accidental (as illustrated below).

The latter two points may be summarised that although a network is stationary, it may not be static. However the stationary nature of the network, but need for possible rapid reconfiguration, may be recognised in setting protocol parameters appropriate to the scenario.

This OLSR implementation has been used as part of a demonstration of a First Generation Unattended Ground Sensor Network by BAE SYSTEMS Avionics Group, Sensor Systems Division and BAE SYSTEMS Advanced Technology Centre. This concept demonstrator was based on earlier mobile network demonstrations using laptops and PDAs (Compaq iPAQs). Figure 4 below shows a demonstration ground sensor node, which contains a low power ARM processor with WLAN, an acoustic interface, a seismic sensor (geo-phone) and a GPS receiver to provide synchronisation and geo-location. Figure 5 below illustrates a camera node that can be connected to a variety of different imager types and controlled via the network; for example the imager can be cued via either the acoustic or seismic sensor nodes.

Figure 4: Sensor Node

Figure 5: Camera Node

The demonstration system architecture also includes a communication gateway that can support requirements for long-haul communication, thus providing connectivity to existing legacy infrastructure. The demonstration network equipment, which has been integrated with an existing in-service sound ranging system as well as operated in a standalone mode, is shown in Figure 6 below.

Figure 6: Network Before Deployment

This network has been demonstrated in trials in both open terrain and urban environments, and shown to provide useful track data for typical battlefield vehicle targets for a number of different deployment scenarios. An example network deployment is shown, to scale, in Figure 7. This shows the links present at one, representative, time as seen by node S0. (This information is taken, through a proprietary management interface, from a combination of OLSR's Neighbour Set, Topology Set and Routing Table.) Links are limited by terrain as well as their usual propagation range, in particular because the antennas are close to the ground.

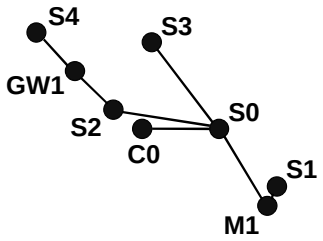

Figure 7: Example Network Deployment

The network was not static however; this may be seen from Table 1 below which shows the mean distance, in hops, of each other node from node S0. The non-integer hop count values show a changing topology. One reason for this is vehicles (including substantial sized military vehicles) moving so as they may block direct paths at various times, but may also include other reasons. This illustrates a need for a protocol which not only configures an ad hoc arrangement of nodes but also maintains a dynamic ad hoc network.

Node	Abbreviation	Distance from S0
Sensor 1	S1	1.74
Sensor 2	S2	1.07
Sensor 3	S3	1.01
Sensor 4	S4	2.75
Controller 0	C0	1.01
Gateway 1	GW1	1.54
Monitor 1	M1	1.02

Table 1: Mean Distances (Number of Hops) from S0

This network successfully maintained connectivity throughout most of the demonstration, as shown in Figure 8 below. There were some problems initially, and a few later dropouts, but throughout most of the trial the network passed all data packets successfully.

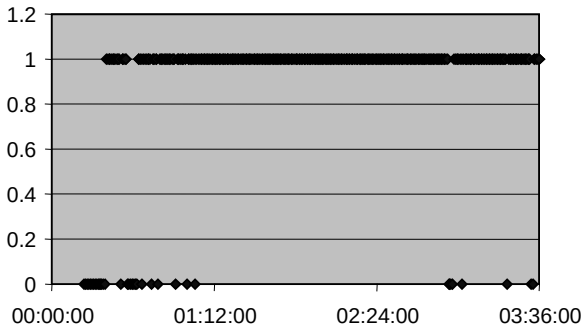

Figure 8: Packet Receptions at S0

Additional work using OLSR, and at least one other ad hoc routing protocol, is ongoing under the collaborative B2NCW (Building Blocks for Network Centric Warfare) programme (see Acknowledgements) which will include autonomous vehicle nodes as well as stationary nodes.

9. Additional Protocol Requirements

OLSR based ad hoc networks may be appropriate for many small scale military and other applications, including, but not limited to, sensor networks. However in considering the requirements for such networks, some additional points may be noted.

First there is one almost universal requirement, although the specifics of the requirement vary from application to application.

- Security.

The following requirements are less universal, although all of interest in at least some applications. Note that these requirements are not independent; the first two are related (and the second to security) as are the last two.

- Low power operation (including power control).
- Covertness (in some cases possibly requiring a reactive protocol).
- Multicast.
- Addressing issues (including IPv6 issues).
- External gateway issues (including aggregation and dynamism).

10. Conclusions

The following key points have been reported in this paper

- The use of a framework facilitates the use of multiple ad hoc routing protocols, and also the portability and extension of even a single protocol.
- BAE SYSTEMS has used this approach to create a comprehensive implementation of OLSR for both simulation purposes and real product application.
- Simulation results have shown the benefits of an extension to include minimum message intervals in a highly mobile scenario.
- The implementation has been used in successful field trials of an ad hoc sensor network which is stationary, but not static, and hence benefits from full mobile ad hoc network functionality.
- Some observations as to additional requirements for ad hoc networks have been made. The most important of these, based on information available to the author, is security.

Acknowledgements

The author gratefully acknowledges the support of his colleagues in BAE SYSTEMS Advanced Technology Centre (especially Mike Otter and Andrew McCabe), BAE SYSTEMS Avionics Group Sensor Systems Division (especially George Matich) and Ericsson Microwave Systems AB and Ericsson Telebit A/S, and the support from the UK, Swedish and Danish MoDs under the EUCLID/Eurofinder programme, Project RTP6.22 (B2NCW).

References

[1] T. Clausen and P. Jacquet (Ed.), "Optimized Link State Routing Protocol (OLSR)", RFC 3626, October 2003.

[2] OPNET Technologies, http://www.opnet.com.

[3] C. Perkins et. al., "Ad hoc On-demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

[4] R. Ogier et. al., "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", RFC 3684, February 2004.

[5] D.B. Johnson et. al., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", draft-ietf-manet-dsr-09.txt, April 2003.

[6] C. Adjih et. al., "Securing the OLSR protocol", 2nd Mediterranean Workshop on Ad Hoc Networks, June 2003.

[7] A. Alexandrescu, "Modern C++ Design: Generic Programming and Design Patterns Applied", Addison Wesley, 2001.

[8] A. McCabe et. al., "Scalability Modelling of Ad Hoc Routing Protocols – A Comparison of OLSR and DSR", 4th Scandinavian Workshop on Ad Hoc Networks, Stockholm, May 2004.