OLSR Developments and Extensions

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

Abstract

Some work which has been undertaken since the last OLSR Interop and Workshop, or which was not discussed there, is presented. First is an implementation of OLSR-encapsulated flooding for multicast and a version of Secure OLSR within the generic software framework which was presented at the last OLSR Interop. Second is a discussion of approaches to the combination of multiple OLSR networks. Both of these cases suggest improvements which could be made to OLSR, however it is also noted that OLSR, as currently defined, is alive and well and usable as it is.

1. Introduction

At the First OLSR Interop and Workshop [1] we presented a summary of an OLSR [2] implementation and application [3]. This paper considers developments since that time, or not reported then, and how they reflect on the future of OLSR.

Two main topics are considered in this paper: an implementation of OLSR-encapsulated flooding for multicast and of Secure OLSR [4] within the implementation framework described in [3], and issues relating to the combination of multiple OLSR networks. In both cases issues arise which suggest improvements which can be made to OLSR.

It should be noted however that, although improvements are possible, OLSR is entirely usable "as is". In [3] we presented some results of the performance of OLSR from trials of real prototype sensor equipment. We do not have any additional results to report here this year, although further trials have taken place. The reason is that the trials have concentrated on application issues, the networking, in particular OLSR, is considered to be sufficiently mature that it can essentially be ignored. This is, we think, a more significant observation than would be a set of performance results of OLSR from such trials.

2. Extensions to OLSR Implementation

In [3] we presented a picture of the BAE Systems OLSR implementation within a general ad hoc routing protocol (AHRP) framework as shown in Figure 1 below. At that time the implementation of AODV [5] and Secure OLSR [4] were hypothetical. The implementation of the OLSR protocol is in class `Olsr`.

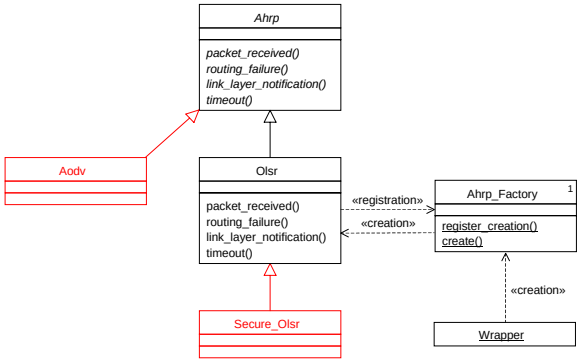

Figure 1: AHRP and OLSR Classes and Creation

We have since implemented a version of Secure OLSR, described in the following section, as a class `Secure_Olsr` derived from class `Olsr`, as shown in Figure 1 above. We have also implemented multicast in the ad hoc network within a similarly derived class `Olsr_Flood` by encapsulating the IP datagrams to be within OLSR messages, using OLSR's optimised flooding mechanism to deliver them to all nodes in the network (or all those within a specified hop count). Nodes which require those datagrams (form part of the appropriate multicast group) can decapsulate them and process them as required.

In common with the rest of the OLSR software, as described in [3], this has used the same code for both implementation and simulation, except for environment specific wrapper code to remove and insert datagrams from and into the IP protocol stack (plus the other services required by standard OLSR).

The interface between class `Olsr` and classes which derive from it, such as class `Secure_Olsr` shown in Figure 1 above, uses the non-virtual interface idiom (NVII) [6] where the base class `Olsr` defines points of customisation of the protocol, which may be inserted by the derived class.

For example the key feature of Secure OLSR is that for each OLSR message there is a matching signature message which cryptographically "guarantees" the OLSR message. Received OLSR messages for which a signature message is not received (within an appropriate time window) are discarded. This may be (and has been) implemented by the object of class `Secure_Olsr` removing all received messages from the base object of class `Olsr`'s storage (via a defined interface) until a matching signalling message and signature message are so removed (together, or separately in either order) whereupon the signalling message can be returned to the object of class `Olsr`, the signature message being retained while still required and later discarded.

This implementation allows most extensions to OLSR implemented in this way to be combined easily. Using C++ virtual inheritance of the derived classes from

© BAE SYSTEMS 2005. All rights reserved..

class `Olsr`, a combined extension can be created by multiple inheritance from the derived classes. Where the extensions over-ride different functions from class `Olsr` no further action is generally needed; when they over-ride the same function a combined function is needed, usually simply calling the appropriate functions in turn.

The class hierarchy, without the object factory shown in Figure 1 above, is now as shown in Figure 2 below.

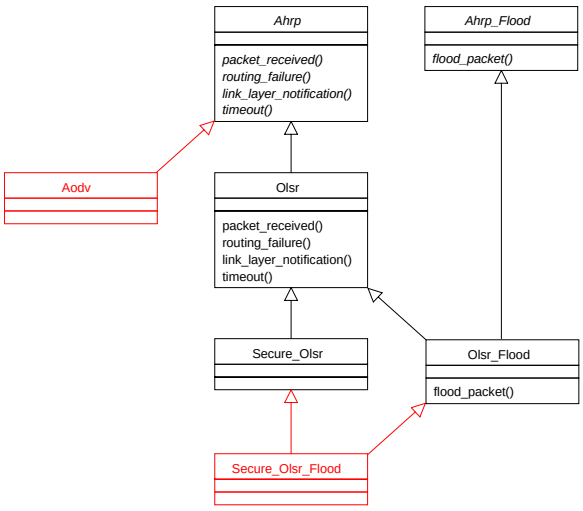

Figure 2: AHRP and OLSR Classes

This hierarchy also shows a new class, `Ahrp_Flood`, which provides a generic interface to a routing protocol which can flood packets, so that the environment specific wrapper code only needs to know about classes `Ahrp` and `Ahrp_Flood` (it can distinguish the cases which cannot flood, such as classes `Olsr` and `Secure_Olsr`, using a C++ dynamic cross-cast). The combined class `Secure_Olsr_Flood` is straightforward to create. Details of the non-virtual interface to class `Olsr`, and as in [3], complete details of the other interfaces shown, are omitted for clarity.

Also not shown in Figure 1 above is a management interface. This allows for the routing protocol to provide information on the network topology to the wrapper for whatever use may be made of it, for example a display of the current state of the network. The interface is proprietary, but defined to be independent of the choice of ad hoc routing protocol. For OLSR it includes information from all of the protocol sets defined in [2], in particular the Link and Topology Sets. Management configuration of the ad hoc routing protocol is currently supported only by dynamic protocol parameter specification, this is protocol specific.

3. Secure OLSR

The implementation of Secure OLSR noted above has followed the approach, and message definitions, of [4], using end to end signature of messages, rather than the hop by hop signature of packets implemented in [7]. The implementation has included timestamp exchange, to prevent replay attacks, as in [4], but has not included key distribution.

Instead the implementation has used Identity Based Encryption (IBE) [8]. This is a private/public key encryption mechanism that differs from conventional public key cryptography (PKC) in that it assigns a known identity to each node, and then each node's public key can be created from its identity using a publicly known function. Then each node can be pre-configured with its identity, private key and this public function, and nodes can then communicate securely without the need for an interactive trusted third party (such as a public key repository). The drawback to this use of IBE is that the nodes must be pre-configured, and by a trusted third party which knows all of the nodes' secret keys (unlike a conventional PKC system where only the node itself knows this). As for a conventional PKC system, but unlike a shared key system, loss or other compromise of a node does not affect the security of other nodes, although to avoid disruption of the network routing the lost node must be repudiated.

Our current implementation of IBE in Secure OLSR uses short key lengths which allow a proof of concept only, practical longer key lengths are under consideration.

An example of a simple scenario which has been demonstrated, using both IPv4 and IPv6, for Secure OLSR is as follows. As shown in Figure 3 below let there be a two hop route from node A to node C, via node B which is an MPR of node C. Traffic (real-time video makes a good demonstration) is sent from A to C via B.

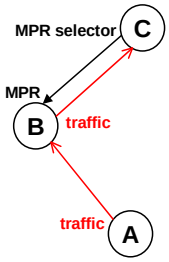

Figure 3: Basic Scenario

Now let node X, which is hostile, arrive within range of nodes A and C, and declare a higher willingness to be an MPR than does node B. Running standard OLSR, node C will elect node X as its MPR in place of node B, and, assuming node X implements OLSR correctly, node A will attempt to route traffic to node C via node X rather than node B. However if node X discards all traffic from node A, the connection to node C is broken, as shown in Figure 4 below and the video display at C stops.

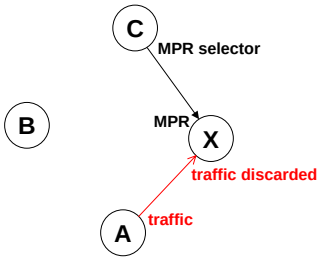

Figure 4: Scenario with Hostile Node

© BAE SYSTEMS 2005. All rights reserved..

Note that this attack is invisible to the OLSR protocol, which is operating as intended. In fact if node X is content to only observe, rather than disrupt, traffic and forward it properly, it can be completely unnoticed. However with Secure OLSR, node X's HELLO messages, with which it attempts to establish itself in the network, are not recognised by either node A or node C, and the route from A to C via B, and its traffic flow (and video display at C) are uninterrupted.

Implementation of Secure OLSR within the software framework described earlier has however highlighted an area where OLSR, although extensible using the definition of new messages as defined in [2], could be improved. Message signatures in Secure OLSR are separate messages to the messages they sign. Because message piggybacking is not mandatory in OLSR (although it is included in our implementation) there can be no guarantee that the message and the signature will be in the same packet even if they are transmitted in the same packet, as forwarding nodes may not maintain the piggybacking.

Consequently the Secure OLSR implementation may receive a packet containing just the signature or the message, and then have to wait for the other. Furthermore it must take care to get certain cases which can occur correct, such as when first a signature is received from a non-MPR selector, then from an MPR selector, then the message is received from the MPR selector, then from the non-MPR selector. In this case the first copy of the message and the second copy of the signature should be those compared, and must be forwarded. Such complications make denial of service attacks easier, adding the need to store messages received as well as possibly increasing the number of decryptions needed, each of which may be an expensive operation.

An improvement would be if the message and signature could be guaranteed to be in the same packet. Although this is an advantage of [7], the end to end signature property is worth retaining. Without mandating piggybacking, which has its own issues, one approach would be if the signature could form part of the same message, rather than being a separate message. (This would also be more efficient in overhead.) Without standardising a specific form of Secure OLSR, which would be premature, this could be accomplished if OLSR were extensible "internally" within messages, as well as "externally" through new messages. This concept is now part of the intended evolution of OLSR into a Standards Track proactive routing protocol [9].

4. Multiple Network OLSR Extensions

An example of a beneficial avenue of development of OLSR, "internal" message extension for message signatures, was considered above. Another example of a desirable OLSR extension, based on a case of interest, may be seen when we consider deploying multiple sensor networks.

Suppose we have two sensor networks, a "blue" network and a "green" network, using separate subnet addresses, and we wish to connect the two. We will assume that they are using distinct physical/data link layers (possibly frequency, or coding) so they do not directly interwork, but that there can be a blue/green node which has an interface in each network, as shown in Figure 5 below.

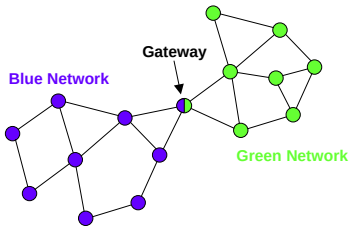

Figure 5: Blue and Green Networks

Using OLSR there are two natural ways to configure the blue/green node to provide the necessary connectivity

- The node can treat these two interfaces as OLSR multiple interfaces and advertise itself to both networks using a MID message. Now in OLSR terms the two networks become a single network; all nodes are aware of all other nodes, in their own or the other network. The drawback with this approach is scalability: if, for example the two networks each have 30 nodes, then the combined network has 59 nodes (allowing for the common node) and must transmit the traffic to match.
- The node can be treated as a gateway between the two networks, advertising each network into the other using an HNA message. Note that the common node now must run two copies of OLSR, one for the blue network, one for the green. This solution solves the scalability problem, the OLSR messages within each network do not pass into the other, and the unnecessary information that each network has about the other in the previous solution is avoided. (A green node does not actually need to know about routes within the blue network, only to go via the blue/green node.)

With two networks there are significant advantages in the latter approach as indicated. However matters become more complicated when we consider having more than one common node or, especially, more than two networks.

With two networks and more than one common node, both solutions continue to work, with the same advantages and disadvantages. However the gateway solution can produce less efficient routes than the multiple interface solution. Because the latter treats everything as one OLSR network, it always produces minimal hop routes. With the gateway approach when, for example, a node in the blue network wants a route to a node in the green network, it may have a choice of gateways. Without knowledge of the details of the green network, and to avoid looping, it must pick the nearest gateway. This may not give the overall shortest route. In addition in the multiple interface solution the shortest route from one blue node to another blue node may go via a green node (and two blue/green nodes). The gateway solution may not find this. In particular if the blue network fragments (owing to node movement,

© BAE SYSTEMS 2005. All rights reserved..

or possibly loss) this may be the only route, and the gateway approach will not find it. Apart however from this final point, which may be tolerable, the possibly increased route lengths in the gateway case must be traded against the reduced signalling overhead, which may be worthwhile.

With more than two networks, the scalability advantage of the gateway solution should begin to assert itself, as, for example four 30 node networks remain that, rather than becoming a single network of almost 120 nodes. Unfortunately however it is precisely here that, with the current definition of OLSR, the gateway solution no longer works.

To start with consider three networks, blue, green and red, with single blue/green and green/red nodes, as shown in Figure 6 below.

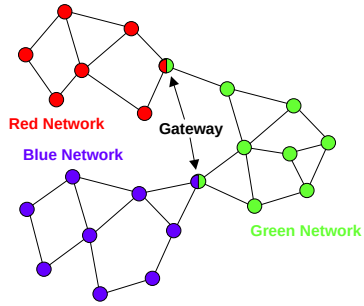

Figure 6: Blue, Green and Red Networks (I)

Now in order that we can route from the blue network to the red network, the blue/green node must advertise not just the green network into the blue network, but also the red network into the blue network. Assuming that this cannot be statically configured, the blue/green node must wait until it is aware of the red network via HNA messages transmitted by the green/red node into the green network. Information indicating the existence of the red network must be transferred from the blue/green node's green network copy of OLSR to its blue network copy OLSR. This transfer could be implemented either by creating some form of direct interface between the two copies of OLSR, or via a shared copy of the routing table. But now from the perspective of the blue network, as advertised by its HNA messages, the blue/green node is a gateway to the green and red networks equally, the fact that the red network is "further away" is lost.

This is not a problem in this case, but one occurs if we also add a blue/red gateway node, as shown in Figure 7 below.

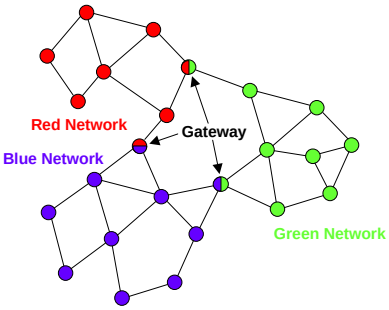

Figure 7: Blue, Green and Red Networks (II)

Now once both the blue/red and blue/green nodes are advertising the red network, a node in the blue network closer to the blue/green gateway will send traffic for the red network to it. Now even if that traffic can now be delivered to the red network, it will have travelled by a definitely inferior route. But it might not even manage that. Once the packets have reached the blue/green gateway, they will set off across the green network. However the blue/green gateway will also advertise the red network to the green network. Consequently, these packets in the will generally be routed back to the blue/green gateway, because it will be closer, and we have a looping problem.

One approach to solve this routing problem is instead of the packets travelling through the green network, they must be tunnelled through it from the blue/green node to the green/red node. This then suggests an additional higher level network, in this case of three nodes, the three gateways. Inter-network traffic travels to a gateway, is routed by tunnelling through intermediate networks to another gateway and then is decapsulated in the destination network through which it travels normally. The higher level network now itself needs a routing algorithm between gateways, the most obvious candidate is another instantiation of OLSR (so that a gateway node now runs three copies). Now discovery of a gateway by receipt of HNA messages in a lower level network becomes a HELLO message in the higher level network.

The use of tunnelling is however unsatisfactory in many regards. The looping problem that demands it could however be solved in another manner if the associated network advertisement had a hop count metric associated with it. For example if the maximum diameter of any MANET under discussion is assumed to be less than some value *N*, then in the example above, with blue green and red networks all connected to each other, the blue/green node should advertise the green network into the blue network with metric zero, and similarly the green/red network should advertise the red network into the green network with metric zero. However when the blue/green node receives an HNA from the green/red network, it should use this to advertise the red network into the blue network with metric *N*. Now, since also the blue/red node will advertise the red network into the blue network with metric zero, traffic from the blue network to the red network will always pick the blue/red node as a gateway, even if closer to the blue/green node (on the assumption that neither is *N* hops or more away). In more complicated scenarios, networks could be advertised with metrics of 2*N*, 3*N* and so on.

There is however some more work needed to produce a complete solution based on this concept. As described, this is a form of proactive distance vector protocol, and suffers from a potential "count to infinity" problem, for example if Figure 6 above is extended to add a yellow network attached to the red network, and then, after metrics are established, the blue network separates from the green network, as shown in Figure 8 below both the green/red node and the red/yellow node will assume that the other is

© BAE SYSTEMS 2005. All rights reserved..

providing the route to the blue network and keep incrementing the other's metric.

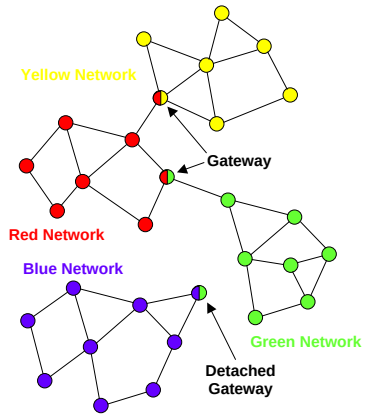

Figure 8: Count to Infinity Scenario

Solutions to this problem have been implemented in various routing protocols (including RIP, DSDV and AODV). It may be noted however that in this case a "lightweight" solution is likely to be satisfactory in many cases, as the number of networks is likely to be limited.

Regardless of the preceding point, this solution requires an extended HNA message, with a metric for each associated network. This could be done, like Secure OLSR, with a separate message, but that is unsafe because if the new message with metrics is in another packet and fails to arrive then the gateways will be assumed to have zero metric, and we can return to the looping scenario. Consequently the metrics should be in the same packet, and hence message. This suggests a modification to the HNA message; however it should be optional because the zero metric case is likely to be the most usual case, and should not have a significant added overhead to allow for the case where metrics are allowed.

This type of scenario also indicates another modification to the HNA specification. Gateways in general, but especially here, may be transient, and a node should be able to advertise that it is no longer supporting an associated network. But current OLSR does not permit this, gateways can only be removed by expiry. This is unlike TC messages, which permit a cancellation option (although there are some minor problems with the current specification) using the ANSN mechanism. Extending this to HNA messages would be straightforward, possibly through the merger of the TC and HNA messages (which would now fundamentally differ only in whether they include a netmask - for efficiency this can be an optional netmask length).

5. Conclusions

This paper presents two messages on OLSR and its future development.

The first is that OLSR is alive and well, and usable either as it is, or using extensions (such as Secure OLSR) supported by the extension mechanism (support for additional message types) already defined. This is evident from real trials and demonstrations.

The second is that there are, nevertheless, areas in which OLSR could be (and is being) improved by supporting "internal" extension of messages as well as "external" extension via new messages. Two such areas are described. One is that Secure OLSR could be significantly simplified by allowing message signatures to be incorporated within messages rather than as separate messages. The other follows from a discussion of joining OLSR networks, and consists of enhancing network advertisement (HNA messages) with added hop count metrics, and the combination of the HNA and TC message concepts.

Acknowledgements

The author gratefully acknowledges the support of his colleagues in BAE SYSTEMS Advanced Technology Centre, collaborators at Ericsson Microwave Systems AB and Ericsson Danmark A/S Telebit, and the support from the UK, Swedish and Danish MoDs under the EUCLID/Eurofinder programme, Project RTP6.22 "Building Blocks for Network Centric Warfare" (B2NCW).

References

[1] First OLSR Interop and Workshop, San Diego, August 2004, http://olsrinterop.free.fr.

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

[3] C.M. Dearlove, "OLSR Simulation, Implementation and Ad Hoc Sensor Network Application", see [1].

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

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

[6] H. Sutter and A. Alexandrescu, "C++ Coding Standards", Addison Wesley, 2004.

[7] Andreas Hafslund et. al., "Secure Extension to the OLSR protocol", see [1].

[8] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing", Proceedings of the 21st Annual Cryptology Conference on Advances in Cryptology, pp. 213-229, Springer Verlag, 2001.

[9] "Next-Gen Proactive MANET Routing", http://www3.ietf.org/proceedings/05mar/slides/manet-6.pdf.

© BAE SYSTEMS 2005. All rights reserved..