Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 30, 2015

J. Yi
LIX, Ecole Polytechnique
B. Parrein
University of Nantes

July 29, 2014

Multi-path Extension for the Optimized Link State Routing Protocol version 2 (OLSRv2)
draft-yi-manet-olsrv2-multipath-02

Abstract

This document specifies a multi-path extension to the Optimized Link State Routing Protocol version 2 (OLSRv2) to discover multiple disjoint paths, so as to improve reliability of the OLSRv2 protocol. The interoperability with OLSRv2 is retained.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The Optimized Link State Routing Protocol version 2 (OLSRv2) [RFC7181] is a proactive link state protocol designed for use in mobile ad hoc networks (MANETs). It generates routing messages periodically to create and maintain a Routing Set, which contains routing information to all the possible destinations in the routing domain. For each destination, there exists an unique Routing Tuple, which indicates the next hop to reach the destination.

This document specifies an extension of the OLSRv2 protocol [RFC7181], to provide multiple disjoint paths for a source-destination pair. Because of the characteristics of MANETs [RFC2501], especially the dynamic topology, having multiple paths is helpful for increasing network throughput, improving transmission reliability and load balancing.

The Multi-path OLSRv2 (MP-OLSRv2) specified in this document uses multi-path Dijkstra algorithm to explore multiple disjoint paths from source to the destination based on the topology information obtained through OLSRv2, and forward the datagrams in a load-balancing manner using source routing. MP-OLSRv2 is designed to be interoperable with OLSRv2.

1.1. Motivation and Experiments to Be Conducted

This document is an experimental extension of OLSRv2 that can increase the data forwarding reliability in dynamic and high-load MANET scenarios by transmitting packet over multiple disjoint paths using source routing. This mechanism is used because:

- o Disjoint paths can avoid single route failure.
- o By having control of that paths at the source, the delay can be provisioned.
- o Certain scenarios require some routers must (or must not) be used.
- o An very important application of this extension is combination with Forward Error Correction coding. This requires disjoint paths. The single path routing is not adapted because the packet drop is normally continuous, in which forward correction coding is not helpful.

While existed deployments, running code and simulations have proven the interest of multipath extension for OLSRv2 in certain networks, more experiments and experiences are still needed to understand the mechanisms of the protocol. The multipath extension for OLSRv2 is expected to be revised and improved to the Standard Track, once sufficient operational experience is obtained. Other than the general experiences including the protocol specification, interoperability with original OLSRv2 implementations, the experiences in the following aspects are highly appreciated:

o Optimal values for the number of multiple paths (NUMBER_OF_PATHS) to be used. This depends on the network topology and router density.

o Optimal values for the cost functions. Cost functions are applied to punish the costs of used links and nodes so as to obtain disjoint paths. What kind of disjointness is desired (node-disjoint or link-disjoint) may depends on the layer 2 protocol used, and can be achieved by setting different sets of cost functions.

o Optimal choice of "key" routers for loose source routing. In some cases, loose source routing is use to reduce overhead or for interoperability with OLSRv2. Other than the basic rules defined in the following of this document, optimal choices of routers to put in the source routing header can be further studied.

o Use of other metric other than hop-count. This multipath extension can be used not only for hop-count metric type, but other metric types that meet the requirement of OLSRv2, such as [I-D.ietf-manet-olsrv2-dat-metric]. The metric type used has also co-relation with the choice of cost functions as indicated in the previous bullet.

o The impacts to the delay variation due to multi-path routing. [RFC2991] brings out some concerns of multi-path routing, especially variable latencies. Although current experiments result show that multi-path routing can reduce the jitter in dynamic scenarios, some transport protocols or applications may be sensitive to the packet re-ordering.

o The disjoint multiple path protocol has interesting application with Forward Error Correction (FEC) Coding, especially for services like video/audio streaming. The combination of FEC coding mechanisms and this extension is thus encouraged.

o In addition to IP source routing based approach, it can be interesting to try multi-path routing in MANET using label-switched flow in the future.

o The usage of multi-topology information. By using [I-D.ietf-manet-olsrv2-multitopology], multiple topologies using different metric types can be obtained. It is encouraged to experiment the use of multiple metrics for building multiple paths also.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses the terminology and notation defined in [RFC5444], [RFC6130], [RFC7181]. Additionally, it defines the following terminology:

OLSRv2 Routing Process - The routing process based on [RFC7181], without multi-path extension specified in this document.

MP-OLSRv2 Routing Process - The routing process based on this specification as an extension to [RFC7181].

3. Applicability Statement

As an extension of OLSRv2, this specification is applicable to MANETs for which OLSRv2 is applicable (see [RFC7181]). It can operate on single, or multiple interfaces, to discover multiple disjoint paths from a source router to a destination router.

MP-OLSRv2 is specially designed for networks with dynamic topology and slow data rate links. By providing multiple paths, higher aggregated bandwidth can be obtained, and the routing process is more robust to packet loss.

In a router supporting MP-OLSRv2, MP-OLSRv2 does not necessarily replace OLSRv2 completely. The extension can be applied for certain applications that are suitable for multi-path routing (mainly video or audio streams), based on the information such as DiifServ Code Point [RFC2474].

Compared to OLSRv2, this extension does not introduce new message type in the air, and is interoperable with OLSRv2 implementations that do not have this extension.

4. Protocol Overview and Functioning

This specification requires OLSRv2 [RFC7181] to:

- o Identify all the reachable routers in the network.
- o Identify a sufficient subset of links in the networks, so that routes can be calculated to all reachable destinations.
- o Provide a Routing Set containing shortest routes from this router to all destinations.

Based on the above information acquired by OLSRv2, the MP-OLSRv2 Routing Process is able to calculate multiple paths to certain destinations based on multi-path Dijkstra algorithm: the Dijkstra algorithm is performed multiple times . In each iteration, the cost of used links are increased (i.e., punished), so that they can be avoided to be chosen in the next iteration. The multi-path Dijkstra algorithm can generate multiple disjoint paths from a source to a destination , and such information is kept in Multi-path Routing Set. The algorithm is invoked on demand, i.e., only when there is data traffic to be sent from the source to the destination, and there is no available routing tuples in the Multi-path Routing Set.

The datagram is forwarded based on source routing. When there is a datagram to be sent to a destination, the source router acquires a path from the Multi-path Routing Set (in Round-Robin fashion here) . The path information is stored in the datagram header as source routing header. The intermediate routers listed in the source routing header (SRH) read the SRH and forward the datagram to the next hop indicated in the SRH.

5. Parameters and Constants

In addition to the parameters and constants defined in [RFC7181], this specification uses the parameters and constants described in this section.

5.1. Router Parameters

NUMBER_OF_PATHS The number of paths desired by the router.

MAX_SRC_HOPS The maximum number of hops allowed to put in the source routing header.

fp Incremental function of multi-path Dijkstra algorithm. It is used to increase costs of links belonging to the previously computed path.

fe Incremental function of multi-path Dijkstra algorithm. It is used to increase costs of links who lead to routers of the previous computed path.

MR_HOLD_TIME It is the minimal time that a Multi-path Routing Tuple SHOULD be kept in the Multi-path Routing Set.

MP_OLSR_HOLD_TIME It is the minimal time that a MP-OLSRv2 Router Tuple SHOULD be kept in the MP-OLSRv2 Router Set.

6. Packets and Messages

This extension employs the routing control messages HELLO and TC (Topology Control) as defined in OLSRv2 [RFC7181]. To support source routing, a source routing header is added to each datagram routed by this extension. Depending on the IP version used, the source routing header is defined in following of this section.

6.1. HELLO and TC messages

HELLO and TC messages used by MP-OLSRv2 Routing Process share the same format as defined in [RFC7181]. In addition, one Message TLV is defined, to identify the originator of the HELLO or TC message is running MP-OLSRv2.

6.1.1. MP_OLSRv2 TLV

An MP_OLSRv2 TLV is a Message TLV that signals the message is generated by an MP-OLSRv2 Routing Process. It does not include any value.

Every HELLO or TC message generated by MP-OLSRv2 Routing Process MUST has one MP_OLSRv2 TLV.

6.2. Datagram

6.2.1. Source Routing Header in IPv4

In IPv4 [RFC0791] networks, the MP-OLSRv2 routing process employs loose source routing, as defined in [RFC0791]. It exists as an option header, with option class 0, and option number 3.

The source route information is kept in the "route data" field of the loss source route header.

6.2.2. Source Routing Header in IPv6

In IPv6 [RFC2460] networks, the MP-OLSRv2 routing process employs the source routing header as defined in [RFC6554], with IPv6 Routing Type 3.

The source route information is kept in the "Addresses" field of the routing header.

7. Information Bases

Each MP-OLSRv2 routing process maintains the information bases as defined in [RFC7181]. Additionally, two more information bases are defined for this specification.

7.1. MP-OLSRv2 Router Set

The MP-OLSRv2 Router Set recordes the routers running the MP-OLSRv2 Routing Process. It consists of MP-OLSRv2 Router Tuples:

(MP_OLSR_addr, MP_OLSR_valid_time)

where:

MP_OLSR_addr - it is the network address of the router that runs MP-OLSRv2 Routing Process;

MP_OLSR_valid_time - it is the time until which the MP-OLSRv2 Router Tuples is considered valid.

7.2. Multi-path Routing Set

The Multi-path Routing Set records the full path information of different paths to the destination. It consists of Multi-path Routing Tuples:

(MR_dest_addr, MR_valid_time, MR_path_set)

where:

MR_dest_addr - it is the network address of the destination, either the network address of an interface of a destination router or the network address of an attached network;

MR_valid_time - it is the time until which the Multi-path Routing Tuples is considered valid;

MP_path_set - it contains the multiple paths to the destination. It consists of Path Tuples.

Each Path Tuple is defined as:

(PT_cost, PT_address[1], PT_address[2], ..., PT_address[n])

where:

PT_cost - the cost of the path to the destination;

PT_address[1...n] - the addresses of intermediate router to be visited numbered from 1 to n.

8. Protocol Details

This protocol is based on OLSRv2, and extended to discover multiple disjoint paths from the source to the destination router. It retains the basic routing control packets formats and processing of OLSRv2 to obtain topology information of the network. The main differences between OLSRv2 routing process is the datagram processing at the source router and datagram forwarding.

8.1. HELLO and TC Message Generation

HELLO and TC messages are generated according to the section 15.1 or section 16.1 of [RFC7181].

A single Message-Type-Specific TLV with Type := MP_OLSRv2 is added to the message.

8.2. HELLO and TC Message Processing

HELLO and TC messages are processed according to the section 15.3 and 16.3 of [RFC7181].

For every HELLO or TC message received, if there exists a TLV with Type := MP_OLSRv2, create or update (if the tuple exists already) the MP-OLSR Router Tuple with

- MP_OLSR_addr = originator of the HELLO or TC message

and set the MP_OLSR_valid_time := current_time + MP_OLSR_HOLD_TIME.

8.3. Datagram Processing at the MP-OLSRv2 Originator

When the MP-OLSRv2 routing process receives a datagram from upper layers or interfaces connecting other routing domains, find the Multi-path Routing Tuple where:

- o MR_dest_addr = destination of the datagram, and
- o MR_valid_time < current_time.

If a matching Multi-path Routing Tuple is found, a Path Tuple is chosen from the MR_path_set in Round-robin fashion (if there are multiple datagrams to be sent). Or else, the Multi-path Dijkstra Algorithm defined in Section 8.4 is invoked, to generate the desired Multi-path Routing Tuple.

The addresses in PT_address[1...n] of the chosen Path Tuple are thus added to the datagram header in order as source routing header, following the rules:

- o Only the addresses exist in MP-OLSR Router Set can be added to the source routing header.
- o If the length of the path (n) is greater than MAX_SRC_HOPS, only the key routers in the path are kept. By default, the key routers are uniformly chosen in the path.
- o The routers with higher priority (such as higher willingness of routing) are preferred.
- o The routers that are considered not appropriate for forwarding indicated by external policies should be avoided.

8.4. Multi-path Dijkstra Algorithm

The Multi-path Dijkstra Algorithm is invoked when there is no available Multi-path Routing Tuple to a desired destination d. The general principle of the algorithm is at step i to look for the shortest path Pi to the destination d. Based on Dijkstra algorithm, the main modification consists in adding two cost functions namely incremental functions fp and fe in order to prevent the next steps to use similar path. fp is used to increase costs of arcs belonging to the previously path Pi (or which opposite arcs belong to it). This encourages future paths to use different arcs but not different vertices. fe is used to increase costs of the arcs who lead to vertices of the previous path Pi. It is possible to choose different fp and fe to get link-disjoint path or node-disjoint routes as necessary. A recommendation of configuration of fp and fe is given in Section 5.

To get NUMBER_OF_PATHS distinct paths, for each path Pi (i = 1, ..., NUMBER_OF_PATHS) do:

1. Run Dijkstra algorithm to get the shortest path Pi for the destination d.

2. Apply cost function fp to the links in Pi.

3. Apply cost function fe to the links who lead to routers used in P.

A simple example of Multi-path Dijkstra Algorithm is illustrated in Appendix A.

By invoking the algorithm depicted above, NUMBER_OF_PATHS distinct paths is obtained, and added to the Multi-path Routing Set, by creating a Multi-path Routing Tuple with:

- MR_dest_addr := destination d

- MR_valid_time := current time + MR_HOLD_TIME

- Each Path Tuple in the MP_path_set corresponds to a path obtained in multi-path Dijkstra algorithm, with PT_cost := cost of the path to the destination d.

8.5. Datagram Forwarding

On receiving a datagram with source routing header, the Destination Address field of the IP header is first compared to the addresses of the local interfaces. If no matching address is found, the datagram is forwarded according OLSRv2 routing process. If a matching local address if found, the datagram is processed as follows:

1. Obtain the next source address Address[i] in the source route header. How to obtain the next source address depends on the IP version used. In IPv4, the position of the next source address is indicated by the "pointer" field of the source routing header [RFC0791]. In IPv6, the position is indicated by "Segments Left" field of the source routing header. If no next source address is found, the forwarding process is finished.

2. Swap Address[i] and destination address in the IP header.

3. Forward the datagram to the destination address according to the OLSRv2 Routing Tuple information through R_local_iface_addr where

   * R_dest_addr = destination address in the IP header

9. Configuration Parameters

This section gives default values and guideline for setting parameters defined in Section 5. Network administrator may wish to change certain, or all the parameters for different network scenarios. As an experimental track protocol, the users of this protocol are also encouraged to explore different parameter setting in various network environments, and provide feedback.

o NUMBER_OF_PATHS = 3. This parameter defines the number of parallel paths used in datagram forwarding. Setting it to one makes the specification identical to OLSRv2. Setting it to too big value can lead to unnecessary computational overhead and inferior paths.

o MAX_SRC_HOPS = 10.

o MR_HOLD_TIME = 10 seconds.

o fp(c) = 4*c, where c is the original cost of the link.

o fe(c) = 2*c, where c is the original cost of the link.

The setting of cost functions fp and fc defines the preference of obtained multiple disjoint paths. If id is the identity functions, 3 cases are possible:

o if id=fe<fp paths tend to be link disjoint;

o if id<fe=fp paths tend to be node-disjoint;

o if id<fe<fp paths also tend to be node-disjoint, but when is not possible they tend to be arc disjoint.

10. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Until April 2014, there are 3 open source implementations of the protocol specified in this document, for both testbed and simulation use.

10.1. Multi-path extension based on nOLSRv2

The implementation is conducted by University of Nantes, France, and is based on Niigata University's nOLSRv2 implementation. It is an open source implementation. The code is available at http://jiaziyi.com/index.php/research-projects/mp-olsr .

It can be used for Qualnet simulations, and be exported to run in testbed. All the specification is implemented in this implementation.

Implementation experience and test data can be found at [ADHOC11].

10.2. Multi-path extension based on olsrd

The implementation is conducted under SEREADMO (Securite des Reseaux Ad Hoc & Mojette) project, and supported by French research agency (RNRT2803). It is based on olsrd (http://www.olsr.org/) implementation, and is open sourced. The code is available at http://jiaziyi.com/index.php/research-projects/sereadmo .

The implementation is for testing the specification in the field. All the specification is implemented in this implementation.

Implementation experience and test data can be found at [ADHOC11].

10.3. Multi-path extension based on umOLSR

The implementation is conducted by University of Nantes, France, and is based on um-olsr implementation (http://masimum.inf.um.es/fjrm/development/um-olsr/). The code is available at http://jiaziyi.com/index.php/research-projects/mp-olsr under GNU GPL license.

The implementation is just for network simulation for NS2 network simulator. All the specification is implemented in this implementation.

Implementation experience and test data can be found at [WCNC08].

11. Security Considerations

This document does currently not specify any security considerations....

12. IANA Considerations

This specification defines two Message TLV Types, which must be allocated from the Message TLV Types repository of [RFC5444].

12.1. HELLO Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-Specific Message TLV for HELLO messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 1.

Type	Description	Allocation Policy
128	MP_OLSRv2	
129-223	Unassigned	Expert Review

Table 1: HELLO Message-Type-specific Message TLV Types

12.2. TC Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-Specific Message TLV for TC messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 2.

Type	Description	Allocation Policy
128	MP_OLSRv2	
129-223	Unassigned	Expert Review

Table 2: TC Message-Type-specific Message TLV Types

13. References

13.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, February 2009.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, March 2012.

[RFC7181] Clausen, T., Dearlove, C., Jacquet, P., and U. Herberg, "The Optimized Link State Routing Protocol Version 2", RFC 7181, April 2014.

13.2. Informative References

[ADHOC11] Yi, J., Adnane, A-H., David, S., and B. Parrein, "Multipath optimized link state routing for mobile ad hoc networks", In Elsevier Ad Hoc Journal, vol.9, n. 1, 28-47, January, 2011.

[I-D.ietf-manet-olsrv2-dat-metric]
Rogge, H. and E. Baccelli, "Packet Sequence Number based directional airtime metric for OLSRv2", draft-ietf-manet-olsrv2-dat-metric-01 (work in progress),

July 2014.

[I-D.ietf-manet-olsrv2-multitopology]
Dearlove, C. and T. Clausen, "Multi-Topology Extension for the Optimized Link State Routing Protocol version 2 (OLSRv2)", draft-ietf-manet-olsrv2-multitopology-04 (work in progress), July 2014.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2501] Corson, M. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[WCNC08] Yi, J., Cizeron, E., Hamma, S., and B. Parrein, "Simulation and performance analysis of MP-OLSR for mobile ad hoc networks", In Proceeding of IEEE Wireless Communications and Networking Conference, 2008.

Appendix A. An example of Multi-path Dijkstra Algorithm

This appendix gives an example of multi-path Dijkstra algorithm. The network topology is depicted in Figure 1.

```
   .-----2-----.
  /     / \     \
 /     /   \     \
1     /     \     5
 \   /       \   /
  \ /         \ /
   3-----------4
```

Figure 1: Network Topology for the on-demand example

The initial cost of all the links is set to 1. The incremental functions fp and fe are defined as fp(c)=4c and fe(c)=2c in this example. Two routes from node 1 to node 5 are demanded.

On the first run of the Dijkstra algorithm, the shortest path 1->2->5 with cost 2 is obtained.

The incremental function fp is applied to increase the cost of the link 1-2 and 2-5, from 1 to 4. fe is applied to increase the cost of the link 1-3, 2-3, 2-4, 4-5, from 1 to 2.

On the second run of the Dijkstra algorithm, the second path 1->3->4->5 with cost 5 is obtained.

Authors' Addresses

Jiazi Yi
LIX, Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33 1 77 57 80 85
Email: jiazi@jiaziyi.com
URI: http://www.jiaziyi.com/

Benoit Parrein
University of Nantes
IRCCyN lab - IVC team
Polytech Nantes, rue Christian Pauc, BP50609
44306 Nantes cedex 3
France

Phone: +33 (0) 240 683 050
Email: Benoit.Parrein@polytech.univ-nantes.fr
URI: http://www.irccyn.ec-nantes.fr/˜parrein