

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 11, 2017

J. Yi
Ecole Polytechnique
B. Parrein
University of Nantes
May 10, 2017

**Multipath Extension for the Optimized Link State Routing Protocol
version 2 (OLSRv2)
draft-ietf-manet-olsrv2-multipath-13**

Abstract

This document specifies a multipath extension for 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

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 11, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Motivation and Experiments to Be Conducted	3
2.	Terminology	5
3.	Applicability Statement	6
4.	Protocol Overview and Functioning	7
5.	Parameters and Constants	8
5.1.	Router Parameters	8
6.	Packets and Messages	8
6.1.	HELLO and TC messages	9
6.1.1.	SOURCE_ROUTE TLV	9
6.2.	Datagram	9
6.2.1.	Source Routing Header in IPv4	9
6.2.2.	Source Routing Header in IPv6	9
7.	Information Bases	10
7.1.	SR-OLSRv2 Router Set	10
7.2.	Multipath Routing Set	10
8.	Protocol Details	11
8.1.	HELLO and TC Message Generation	11
8.2.	HELLO and TC Message Processing	11
8.3.	MPR Selection	12
8.4.	Datagram Processing at the MP-OLSRv2 Originator	12
8.5.	Multipath Calculation	14
8.5.1.	Requirements of Multipath Calculation	14
8.5.2.	Multipath Dijkstra Algorithm	15
8.6.	Multipath Routing Set Updates	16
8.7.	Datagram Forwarding	16
9.	Configuration Parameters	17
10.	Implementation Status	18
10.1.	Multipath extension based on nOLSRv2	18
10.2.	Multipath extension based on olsrd	18
10.3.	Multipath extension based on umOLSR	19
11.	Security Considerations	19
12.	IANA Considerations	20
12.1.	Message TLV Types	20
13.	Acknowledgments	21
14.	References	21
14.1.	Normative References	21
14.2.	Informative References	22
Appendix A.	Examples of Multipath Dijkstra Algorithm	23
Authors' Addresses	25

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 a 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 when appropriate 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 forwarding reliability, and load balancing.

Multipath OLSRV2 (MP-OLSRv2) specified in this document uses the Multipath Dijkstra algorithm by default to explore multiple disjoint paths from a source router to a destination router based on the topology information obtained through OLSRV2, and to 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 datagrams over multiple disjoint paths using source routing. This mechanism is used because:

- o Disjoint paths can avoid single route failures.
- o Transmitting datagrams through parallel paths can increase aggregated throughput.
- o Some scenarios may require some routers must (or must not) be used.
- o Having control of the paths at the source benefits the load balancing and traffic engineering.
- o An application of this extension is in combination with Forward Error Correction (FEC) coding applied across packets (erasure coding) [[WPMC11](#)]. Because the packet drop is normally bursty in a path (for example, due to route failure), erasure coding is less effective in single path routing protocols. By providing multiple disjoint paths, the application of erasure coding with multipath

protocol is more resilient to routing failures.

While in existing 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 effects of the protocol. The multipath extension for OLSRV2 is expected to be revised and improved to the Standards Track once sufficient operational experience is obtained. Other than general experiences including the protocol specification and interoperability with base OLSRV2 implementations, the experiences in the following aspects are highly appreciated:

- o Optimal values for the number of multiple paths (NUMBER_OF_PATHS, [Section 5](#)) to be used. This depends on the network topology and router density.
- o Optimal values used in the metric functions. Metric functions are applied to increase the metric of used links and nodes so as to obtain disjoint paths. What kind of disjointness is desired (node-disjoint or link-disjoint) may depend on the layer 2 protocol used, and can be achieved by applying different sets of metric functions.
- o Use of different metric types. This multipath extension can be used with metric types that meet the requirement of OLSRV2, such as [\[RFC7779\]](#). The metric type used has also impact to the choice of metric functions as indicated in the previous bullet point.
- o The impact of partial topology information to multipath calculation. OLSRV2 maintains a partial topology information base to reduce protocol overhead. Although with existing experience, multiple paths can be obtained even with such partial information, the calculation might be impacted, depending on the Multi-Point Relay (MPR) selection algorithm used.
- o Use of IPv6 loose source routing. In the current specification, only strict source routing is used for IPv6 based on [\[RFC6554\]](#). In [\[I-D.ietf-6man-segment-routing-header\]](#), the use of the loose source routing is also proposed in IPv6. In scenarios where the length of the source routing header is critical, the loose source routing can be considered.
- o Optimal choice of "key" routers for loose source routing. In some cases, loose source routing is used to reduce overhead or for interoperability with OLSRV2 routers. Other than the basic rules defined in the following parts of this document, optimal choices of routers to put in the loose source routing header can be further studied.

- o Different path-selection schedulers. By default, round-robin scheduling is used to select a path to be used for datagrams. In some scenarios, weighted scheduling can be considered: for example, the paths with lower metrics (i.e., higher quality) can transfer more datagrams compared to paths with higher metrics.
- o The impacts of the delay variation due to multipath routing. [\[RFC2991\]](#) brings out some concerns of multipath routing, especially variable latencies. Although current experiment results show that multipath routing can reduce the jitter in dynamic scenarios, some transport protocols or applications may be sensitive to the datagram re-ordering.
- o The disjoint multipath protocol has interesting application with erasure coding, especially for services like video/audio streaming [\[WPMC11\]](#). The combination of erasure coding mechanisms and this extension is thus encouraged.
- o Different algorithms to obtain multiple paths, other than the default Multipath Dijkstra algorithm introduced in this specification.
- o The use of multi-topology information. By using [\[RFC7722\]](#), multiple topologies using different metric types can be obtained. Although there is no work defining how this extension can make use of the multi-topology information base yet, it is encouraged to experiment with the use of multiple metrics for building multiple paths.

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 - A routing process based on [\[RFC7181\]](#), without multipath extension specified in this document.

MP-OLSRV2 Routing Process - A multipath routing process based on this specification as an extension to [\[RFC7181\]](#).

SR-OLSRV2 Routing Process - A OLSRV2 Routing Process that supports source routing, or an MP-OLSRV2 Routing Process.

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 designed for networks with dynamic topology by avoiding single route failure. It can also provide higher aggregated throughput and load balancing.

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 multipath routing (mainly video or audio streams), based on information such as a DiffServ codepoint [[RFC2474](#)].

Compared to OLSRV2, this extension does not introduce any new message type. A new Message TLV Type is introduced to identify the routers that support forwarding based on source routing header. It is interoperable with OLSRV2 implementations that do not have this extension: as the MP-OLSRV2 uses source routing, in IPv4 networks the interoperability is achieved by using the loose source routing header; in IPv6 networks, it is achieved by eliminating routers that do not support IPv6 strict source routing.

MP-OLSRV2 supports two different, but interoperable multipath calculation approaches: proactive and reactive. In the proactive calculation, the paths to all the destinations are calculated before needed. In the reactive calculation, only the paths to desired destination(s) are calculated on demand. The proactive approach requires more computational resources than the reactive one. The reactive approach requires the IP forwarding plane to trigger the multipath calculation.

MP-OLSRV2 forwards datagrams using the source routing header. As there are multiple paths to each destination, MP-OLSRV2 requires the IP forwarding plane to be able to choose which source route to be put in the source routing header based on the path scheduler defined by MP-OLSRV2. For IPv4 networks, implementation of loose source routing is required following [[RFC0791](#)]. For IPv6 networks, implementation of strict source routing is required following the source routing header generation and processing defined in [[RFC6554](#)].

4. Protocol Overview and Functioning

This specification uses 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 the shortest routes from this router to all destinations.

In addition, the MP-OLSRV2 Routing Process identifies the routers that support source routing by adding a new Message TLV in HELLO and TC messages. Based on the above information acquired, every MP-OLSRV2 Routing Process is aware of a reduced topology map of the network and the routers supporting source routing.

A Multipath Routing Set containing the multipath information is maintained. It may either be proactively calculated or reactively calculated:

- o In the proactive approach, multiple paths to all possible destinations are calculated and updated based on control message exchange. The routes are thus available before they are actually needed.
- o In the reactive approach, a multipath algorithm is invoked on demand, i.e., only when there is a datagram to be sent from the source to the destination, and there is no available Routing Tuple in the Multipath Routing Set. This requires the IP forwarding information base to trigger the multipath calculation specified in [Section 8.5](#) when no Multipath Routing Tuple is available. The reactive operation is local in the router and no additional routing control messages exchange is required. When the paths are being calculated, the datagrams SHOULD be buffered unless the router does not have enough memory.

Routers in the same network may choose either proactive or reactive multipath calculation independently according to their computation resources. The Multipath Dijkstra algorithm (defined in [Section 8.5](#)) is introduced as the default algorithm to generate multiple disjoint paths from a source to a destination, and such information is kept in the Multipath 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 Multipath Routing Set (MAY be round-robin, or other

scheduling algorithms). The path information is stored in the datagram header using the source routing header.

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 be put in the source routing header. A value set zero means there is no limitation on the maximum number of hops. In an IPv6 network, it MUST be set to 0 because [\[RFC6554\]](#) supports only strict source routing. All the intermediate routers MUST be included in the source routing header, which makes the number of hops to be kept a variable. In an IPv4 network, it MUST be strictly less than 11 and greater than 0 due to the length limit of the IPv4 header.

CUTOFF_RATIO The ratio that defines the maximum metric of a path compared to the shortest path kept in the OLSRV2 Routing Set. For example, the metric to a destination is R_metric based on the Routing Set. Then the maximum metric allowed for a path is $CUTOFF_RATIO * R_metric$. **CUTOFF_RATIO** MUST be greater than or equal to 1. Note that setting the value to 1 means looking for equal length paths, which may not be possible in some networks.

SR_TC_INTERVAL The maximum time between the transmission of two successive TC messages by an MP-OLSRv2 Routing Process.

SR_HOLD_TIME The minimum value in the TLV with Type = **VALIDITY_TIME** included in TC messages generated based on **SR_TC_INTERVAL**.

6. Packets and Messages

This extension employs the routing control messages HELLO and TC (Topology Control) as defined in OLSRV2 [\[RFC7181\]](#) to obtain network topology information. For the datagram 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 this section.

6.1. HELLO and TC messages

HELLO and TC messages used by the MP-OLSRv2 Routing Process use the same format as defined in [[RFC7181](#)]. In addition, a new Message TLV type is defined, to identify the originator of the HELLO or TC message that supports source route forwarding. The new Message TLV type is introduced for enabling MP-OLSRv2 as an extension of OLSRv2: only the routers supporting source-route forwarding can be used in the source routing header of a datagram, because adding a router that does not understand the source routing header will cause routing failure.

6.1.1. SOURCE_ROUTE TLV

SOURCE_ROUTE TLV is a Message TLV signaling that the message is generated by a router that supports source-route forwarding. It can be an MP-OLSRv2 Routing Process, or an OLSRv2 Routing Process that supports source-route forwarding.

Every HELLO or TC message generated by a MP-OLSRv2 Routing Process MUST have exactly one SOURCE_ROUTE TLV without value.

Every HELLO or TC message generated by an OLSRv2 Routing Process MUST have exactly one SOURCE_ROUTE TLV, if the OLSRv2 Routing Process supports source-route forwarding, and is willing to join the source route generated by other MP-OLSRv2 Routing Processes. The existence of SOURCE_ROUTE TLV MUST be consistent for a specific OLSRv2 Routing Process, i.e., either it adds SOURCE_ROUTE TLV to all its HELLO/TC messages, or it does not add SOURCE_ROUTE TLV to any HELLO/TC messages.

6.2. Datagram

6.2.1. Source Routing Header in IPv4

In IPv4 [[RFC0791](#)] networks, the MP-OLSRv2 Routing Process employs the loose source routing header, 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 loose 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 [section 3 of \[\[RFC6554\]\(#\)\]](#), 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, a Multipath Information Base is used for this specification. It includes the protocol sets as defined below.

7.1. SR-OLSRv2 Router Set

The SR-OLSRv2 Router Set records the routers that support source-route forwarding. This includes routers that run the MP-OLSRv2 Routing Process or the OLSRV2 Routing Process with source-route forwarding support. The set consists of SR-OLSRv2 Router Tuples:

(SR_addr, SR_time)

where:

SR_addr - is the original address of the router that supports source-route forwarding;

SR_time - is the time until which the SR-OLSRv2 Router Tuple is considered valid.

7.2. Multipath Routing Set

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

(MR_dest_addr, MR_path_set)

where:

MR_dest_addr - 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;

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

Each Path Tuple is defined as:

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

where:

PT_metric - is the metric of the path to the destination, measured in LINK_METRIC_TYPE defined in [\[RFC7181\]](#);

PT_address[1, ..., n-1] - are the addresses of intermediate routers to be visited numbered from 1 to n-1, where n is the number of routers in the path, i.e., the hop count.

8. Protocol Details

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

8.1. HELLO and TC Message Generation

HELLO messages are generated according to [Section 15.1 of \[RFC7181\]](#), plus a single message TLV with Type := SOURCE_ROUTE included.

TC message are generated according to [Section 16.1 of \[RFC7181\]](#) plus a single message TLV with Type := SOURCE_ROUTE included. At least one TC message MUST be generated by an MP-OLSRv2 Routing Process during the SR_TC_INTERVAL ([Section 5](#)), which is greater than TC_INTERVAL.

TC message generation based on SR_TC_INTERVAL does not replace the ordinary TC message generation specified in [\[RFC7181\]](#) and MUST NOT carry any advertised neighbor addresses. This is due to the fact that not all routers will generate TC messages based on OLSRv2. The TC generation based on SR_TC_INTERVAL serves for those routers to advertise the SOURCE_ROUTE TLV so that the other routers can be aware of the source-route enabled routers so as to be used as destinations of multipath routing. The validity time associated with the VALIDITY_TIME TLV in such TC messages equals SR_HOLD_TIME, which MUST be greater than the SR_TC_INTERVAL.

8.2. HELLO and TC Message Processing

HELLO and TC messages are processed according to [section 15.3](#) and [16.3 of \[RFC7181\]](#).

In addition to the reasons specified in [\[RFC7181\]](#) for discarding a HELLO message or a TC message on reception, a HELLO or TC message

received MUST be discarded if it has more than one Message TLV with Type = SOURCE_ROUTE.

For every HELLO or TC message received, if there is a Message TLV with Type := SOURCE_ROUTE, create or update (if the Tuple exists already) the SR-OLSR Router Tuple with

- o SR_addr := originator address of the HELLO or TC message
- o SR_time := current_time + validity time of the TC or HELLO message defined in [\[RFC7181\]](#), unless the existing SR_time is greater than the newly calculated the SR_time.

8.3. MPR Selection

Each MP-OLSRv2 Routing Process selects routing MPRs and flooding MPRs following [Section 18 of \[RFC7181\]](#). In a mixed network with OLSRv2-only routers, the following considerations apply when calculating MPRs:

- o MP-OLSRv2 routers SHOULD be preferred as routing MPRs to increase the possibility of finding disjoint paths using MP-OLSRv2 routers.
- o The number of routing MPRs that run MP-OLSRv2 Routing Process MUST be equal or greater than NUMBER_OF_PATHS if there are enough MP-OLSRv2 symmetric neighbors. Otherwise all the MP-OLSRv2 routers are selected as routing MPRs, except the routers with willingness WILL_NEVER.

8.4. Datagram Processing at the MP-OLSRv2 Originator

If datagrams without source routing header need to be forwarded using multiple paths (for example, based on the information of a DiffServ codepoint [\[RFC2474\]](#)), the MP-OLSRv2 Routing Process will try to find the Multipath Routing Tuple where:

- o MR_dest_addr = destination of the datagram

If no matching Multipath Routing Tuple is found and the Multipath Routing Set is maintained proactively, it indicates that there is no multipath route available to the desired destination. The datagram is forwarded following the OLSRv2 Routing Process.

If no matching Multipath Routing Tuple is found and the Multipath Routing Set is maintained reactively, the multipath algorithm defined in [Section 8.5](#) is invoked, to calculate the Multipath Routing Tuple to the destination. If the calculation does not return any Multipath Routing Tuple, the following steps are aborted and the datagram is

forwarded following the OLSRV2 Routing Process.

If a matching Multipath Routing Tuple is obtained, the Path Tuples of the Multipath Routing Tuple are applied to the datagrams using round-robin scheduling. For example, there are 2 path Tuples (Path-1, Path-2) for destination router D. A series of datagrams (Packet-1, Packet-2, Packet-3, ... etc.) are to be sent router D. Path-1 is then chosen for Packet-1, Path-2 for Packet-2, Path-1 for Packet 3, etc. Other path scheduling mechanisms are also possible and will not impact the interoperability of different implementations.

The addresses in PT_address[1, ..., n-1] of the chosen Path Tuple are thus added to the datagram header as the source routing header. For IPv6 networks, strict source routing is used, thus all the intermediate routers in the path are stored in the source routing header following the format defined in [section 3 of \[RFC6554\]](#) with Routing Type set to 3.

For IPv4 networks, loose source routing is used, with the following rules:

- o Only the addresses that exist in SR-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 ([Section 5](#)) or adding the whole path information exceeds the MTU, only the "key" routers in the path are kept. By default, the key routers are uniformly chosen in the path. If further information such as capacity of the routers (e.g., battery life) or the routers' willingness in forwarding data is available, the routers with higher capacity and willingness are preferred.
- o The routers that are considered not appropriate for forwarding indicated by external policies should be avoided.

It is not recommended to fragment the IP packet if the packet with the source routing header would exceed the minimum MTU along the path. Depending on the size of the routing domain, the MTU should be at least $1280 + 40$ (for the outer IP header) + $16 * \text{diameter of the network in number of hops}$ (for the source routing header). If the links in the network have different MTU sizes, by using technologies like Path MTU Discovery, the routers are able to be aware of the MTU along the path. The size of the datagram plus the size of IP headers (including the source routing header) should not exceed the minimum MTU along the path, otherwise, the source routing should not be used.

If the destination of the datagrams is out the MP-OLSRv2 routing domain, the datagram must be source route to the gateway between the

MP-OLSRV2 routing domain and the rest of the Internet. The gateway MUST remove the source routing header before forwarding the datagram to the rest of the Internet.

8.5. Multipath Calculation

8.5.1. Requirements of Multipath Calculation

The Multipath Routing Set maintains the information of multiple paths to the destination. The Path Tuples of the Multipath Routing Set ([Section 7.2](#)) are generated based on a multipath algorithm.

For each path to a destination, the algorithm must provide:

- o The metric of the path to the destination,
- o The list of intermediate routers on the path.

For IPv6 networks, as strict source routing is used, only the routers that exist in the SR-OLSRv2 Router Set are considered in the path calculation, i.e., only the source-routing supported routers can exist in the path.

After the calculation of multiple paths, the metric of paths (denoted c_i for path i) to the destination is compared to the R_metric of the the OLSRV2 Routing Tuple ([\[RFC7181\]](#)) to the same destination. If the metric c_i is greater than $R_metric * CUTOFF_RATIO$ ([Section 5](#)), the corresponding path i SHOULD NOT be used. If less than 2 paths are found with metrics less than $R_metric * CUTOFF_RATIO$, the router SHOULD fall back to OLSRV2 Routing Process without using multipath routing. This can happen if there are too many OLSRV2-only routers in the network, and requiring multipath routing may result in inferior paths.

By invoking the multipath algorithm, up to `NUMBER_OF_PATHS` paths are obtained and added to the Multipath Routing Set by creating a Multipath Routing Tuple with:

- o `MR_dest_addr` := destination of the datagram
- o An `MP_path_set` with calculated Path Tuples. Each Path Tuple corresponds to a path obtained in the Multipath Dijkstra algorithm, with `PT_metric` := metric of the calculated path and `PT_address[1, ..., n-1]` := list of intermediate routers.

8.5.2. Multipath Dijkstra Algorithm

This section introduces the Multipath Dijkstra Algorithm as a default algorithm. It tries to obtain disjoint paths when appropriate, but does not guarantee strict disjoint paths. The use of other algorithms is not prohibited, as long as the requirements described in [Section 8.5.1](#) are met. Using different multipath algorithms will not impact the interoperability.

The general principle of the Multipath Dijkstra Algorithm [[ADHOC11](#)] is using Dijkstra algorithm for multiple iterations, and at iteration i to look for the shortest path $P[i]$ to the destination d . After each iteration, the metric of used links is increased. Compared to the original Dijkstra's algorithm, the main modification consists in adding two incremental functions named metric functions fp and fe in order to prevent the next steps resulting in similar paths:

- o $fp(c)$ is used to increase metrics of arcs belonging to the previous path $P[i-1]$ (with $i > 1$), where c is the value of the previous metric. This encourages future paths to use different arcs but not different vertices.
- o $fe(c)$ is used to increase metrics of the arcs that lead to intermediate vertices of the previous path $P[i-1]$ (with $i > 1$), where c is the value of the previous metric. The "lead to" means that only one vertex of the arc belongs to the previous path $P[i-1]$, while the other vertex does not. The "intermediate" means that the source and destination vertices are not considered.

Considering the simple example in Figure 1: a path $P[i]$ S--A--D is obtained at step i . For the next step, the metric of link S--A and A--D are to be increased using $fp(c)$, because they belong to the path $P[i]$. A--B is to be increased using $fe(c)$, because A is an intermediate vertex of path $P[i]$, and B is not part of $P[i]$. B--D is unchanged.

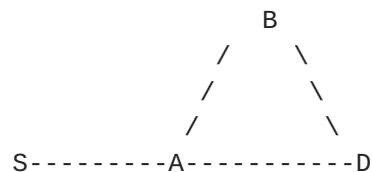


Figure 1

It is possible to choose different fp and fe to get link-disjoint paths or node-disjoint paths as desired. A recommendation for

configuration of fp and fe is given in [Section 9](#).

To get NUMBER_OF_PATHS different paths, for each path P[i] (i = 1, ..., NUMBER_OF_PATHS) do:

1. Run Dijkstra's algorithm to get the shortest path P[i] for the destination d.
2. Apply metric function fp to the metric of links (in both directions) in P[i].
3. Apply metric function fe to the metric of links (in both directions) that lead to routers used in P[i].

A simple example of the Multipath Dijkstra Algorithm is illustrated in [Appendix A](#).

8.6. Multipath Routing Set Updates

The Multipath Routing Set MUST be updated when the Local Information Base, the Neighborhood Information Base, or the Topology Information Base indicate a change (including of any potentially used outgoing neighbor metric values) of the known symmetric links and/or attached networks in the MANET, hence changing the Topology Graph, as described in [section 17.7 of \[RFC7181\]](#). How the Multipath Routing Set is updated depends on whether the set is maintained reactively or proactively:

- o In reactive mode, all the Tuples in the Multipath Routing Set are removed. The new arriving datagrams will be processed as specified in [Section 8.4](#);
- o In proactive mode, the route to all the destinations are updated according to [Section 8.5](#).

8.7. Datagram Forwarding

In IPv4 networks, datagrams are forwarded using loose source routing as specified in [Section 3.1 of \[RFC0791\]](#).

In IPv6 networks, datagrams are forwarded using strict source routing as specified in [Section 4.2 of \[RFC6554\]](#), except the applied routers are MP-OLSRv2 routers rather than RPL routers. The last hop of the source route MUST remove the source routing header.

9. Configuration Parameters

This section gives default values and guidelines for setting parameters defined in [Section 5](#). Network administrators may wish to change certain or all the parameters for different network scenarios. As an experimental 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 large values may lead to unnecessary computational overhead and inferior paths.
- o `MAX_SRC_HOPS := 10`, for IPv4 networks. For IPv6 networks, it MUST be set to 0, i.e., no constraint on maximum number of hops.
- o `CUTOFF_RATIO := 1.5`. It MUST be greater or equal than 1.
- o `SR_TC_INTERVAL := 10 x TC_INTERVAL`. It SHOULD be significantly greater than `TC_INTERVAL` to reduce unnecessary TC message generations.
- o `SR_HOLD_TIME := 32 x TC_INTERVAL`. It MUST be greater than `SR_TC_INTERVAL`. It SHOULD be greater than `30 x TC_INTERVAL`.

If Multipath Dijkstra Algorithm is applied:

- o `fp(c) := 4*c`, where `c` is the original metric of the link.
- o `fe(c) := 2*c`, where `c` is the original metric of the link.

The setting of metric functions `fp` and `fc` defines the preference of obtained multiple disjoint paths. If `id` is the identity function, i.e., `fp(c)=c`, 3 cases are possible:

- o if `id=fe<fp`: only increase the metric of related links;
- o if `id<fe=fp`: apply equal increase to the metric of related nodes and links;
- o if `id<fe<fp`: apply greater increase to the metric of related links.

Increasing the metric of related links or nodes means avoiding the use of such links or nodes in the next path to be calculated.

10. Implementation Status

The RFC Editor is advised to remove the entire section before publication, as well as the reference to [RFC 7942](#).

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 [[RFC7942](#)]. 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 [[RFC7942](#)], "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 2015, there are 3 open source implementations of the protocol specified in this document, for both testbed and simulation use.

10.1. Multipath 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 https://github.com/yijiazi/mpolsr_qualnet and <http://jiiaziyi.com/index.php/research-projects/mp-olsr> .

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

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

10.2. Multipath 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 https://github.com/yijiazi/mpolsr_testbed and <http://jiiaziyi.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] and [GIIS14].

10.3. Multipath 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 https://github.com/yijiazi/mpolsr_ns2 and <http://jiiaziyi.com/index.php/research-projects/mp-olsr> under GNU GPL license.

The implementation is 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

As an extension of [RFC7181], the security considerations and security architecture illustrated in [RFC7181] are applicable to this MP-OLSRv2 specification. The implementations without security mechanisms are vulnerable to threats discussed in [I-D.ietf-manet-olsrv2-sec-threats].

In a mixed network with OLSRv2-only routers, a compromised router can add SOURCE_ROUTE TLVs in its TC and HELLO messages, which will make other MP-OLSRv2 Routing Processes believe that it supports source routing. This will increase the possibility of being chosen as MPRs and put into the source routing header. The former will make it possible to manipulate the flooding of TC messages and the latter will make the datagram pass through the compromised router.

As with [RFC7181], a conformant implementation of MP-OLSRv2 MUST, at minimum, implement the security mechanisms specified in [RFC7183] to provide integrity and replay protection of routing control messages.

The MP-OLSRv2 Routing Process MUST drop datagrams entering or exiting a OLSRv2/MP-OLSRv2 routing domain that contain a source routing

header. Compared to OLSRv2, the use of the source routing header in this specification introduces vulnerabilities related to source routing attacks, which include bypassing filtering devices, bandwidth exhaustion of certain routers, etc. Those attacks are discussed in [Section 5 of \[RFC6554\]](#) and [\[RFC5095\]](#). The influence is limited to the OLSRv2/MP-OLSRv2 routing domain, because the source routing header is used only in the current routing domain.

If the multiple paths are calculated reactively, the datagrams SHOULD be buffered while the paths are being calculated. Because the path calculation is local and no control message is exchanged, the buffering time should be trivial. However, depending on the CPU power and memory of the router, a maximum buffer size SHOULD be set to avoid occupying too much memory of the router. When the buffer is full, the oldest datagrams are dropped. A possible attack that a malicious application could launch is that it initiates a large amount of datagrams to all the other routers in the network, thus triggering path calculation to all the other routers and during which the datagrams are buffered. This might flush other legitimate datagrams. But the impact of the attack is transient: once the path calculation is finished, the datagrams are forwarded and the buffer goes back to empty.

12. IANA Considerations

This section adds one new Message TLV, allocated as a new Type Extension to an existing Message TLV.

12.1. Message TLV Types

This specification updates the Message Type 7 by adding the new Type Extension SOURCE_ROUTE, as illustrated in Table 1.

Type	Name	Description	Reference
Extension			
TBD	SOURCE_ROUTE	Indicates that the originator of the message supports source route forwarding. No value.	This specification

Table 1: SOURCE_ROUTE type for [RFC 5444](#) Type 7 Message TLV Type Extensions

13. Acknowledgments

The authors would like to thank Sylvain David, Asmaa Adnane, Eddy Cizeron, Salima Hamma, Pascal Lesage and Xavier Lecourtier for their efforts in developing, implementing and testing the specification. The authors also appreciate valuable discussions with Thomas Clausen, Ulrich Herberg, Justin Dean, Geoff Ladwig, Henning Rogge, Marcus Barkowsky and especially Christopher Dearlove for his multiple rounds of reviews during the working group last calls.

14. References

14.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", [RFC 5444](#), DOI 10.17487/RFC5444, February 2009, <<http://www.rfc-editor.org/info/rfc5444>>.
- [RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", [RFC 6130](#), DOI 10.17487/RFC6130, April 2011, <<http://www.rfc-editor.org/info/rfc6130>>.
- [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](#), DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.
- [RFC7181] Clausen, T., Dearlove, C., Jacquet, P., and U. Herberg, "The Optimized Link State Routing Protocol Version 2", [RFC 7181](#), DOI 10.17487/RFC7181, April 2014, <<http://www.rfc-editor.org/info/rfc7181>>.
- [RFC7183] Herberg, U., Dearlove, C., and T. Clausen, "Integrity Protection for the Neighborhood Discovery Protocol (NHDP) and Optimized Link State Routing Protocol Version 2

(OLSRv2)", [RFC 7183](#), DOI 10.17487/RFC7183, April 2014, <<http://www.rfc-editor.org/info/rfc7183>>.

14.2. 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.
- [GIIS14] Macedo, R., Melo, R., Santos, A., and M. Nogueira, "Experimental performance comparison of single-path and multipath routing in VANETs", In Global Information Infrastructure and Networking Symposium (GIIS), 2014 , vol. 1, no. 6, pp. 15-19, 2014.
- [I-D.ietf-6man-segment-routing-header] Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", [draft-ietf-6man-segment-routing-header-06](#) (work in progress), March 2017.
- [I-D.ietf-manet-olsrv2-sec-threats] Clausen, T., Herberg, U., and J. Yi, "Security Threats to the Optimized Link State Routing Protocol version 2 (OLSRv2)", [draft-ietf-manet-olsrv2-sec-threats-04](#) (work in progress), January 2017.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [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](#), DOI 10.17487/RFC2474, December 1998, <<http://www.rfc-editor.org/info/rfc2474>>.
- [RFC2501] Corson, S. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", [RFC 2501](#), DOI 10.17487/RFC2501, January 1999, <<http://www.rfc-editor.org/info/rfc2501>>.
- [RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and

- Multicast Next-Hop Selection", [RFC 2991](#), DOI 10.17487/[RFC2991](#), November 2000,
<<http://www.rfc-editor.org/info/rfc2991>>.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", [RFC 5095](#), DOI 10.17487/RFC5095, December 2007,
<<http://www.rfc-editor.org/info/rfc5095>>.
- [RFC7722] Dearlove, C. and T. Clausen, "Multi-Topology Extension for the Optimized Link State Routing Protocol Version 2 (OLSRv2)", [RFC 7722](#), DOI 10.17487/RFC7722, December 2015,
<<http://www.rfc-editor.org/info/rfc7722>>.
- [RFC7779] Rogge, H. and E. Baccelli, "Directional Airtime Metric Based on Packet Sequence Numbers for Optimized Link State Routing Version 2 (OLSRv2)", [RFC 7779](#), DOI 10.17487/[RFC7779](#), April 2016,
<<http://www.rfc-editor.org/info/rfc7779>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", [BCP 205](#), [RFC 7942](#), DOI 10.17487/RFC7942, July 2016,
<<http://www.rfc-editor.org/info/rfc7942>>.
- [WCNC08] Yi, J., Cizeron, E., Hama, 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.
- [WPMC11] Yi, J., Parrein, B., and D. Radu, "Multipath routing protocol for manet: Application to H.264/SVC video content delivery", In Proceeding of 14th International Symposium on Wireless Personal Multimedia Communications.

[Appendix A.](#) Examples of Multipath Dijkstra Algorithm

This appendix gives two examples of Multipath Dijkstra algorithm.

A network topology is depicted in Figure 2.

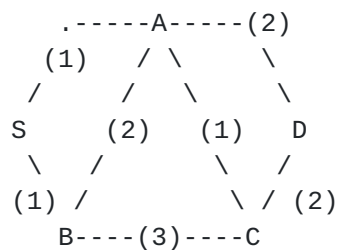


Figure 2

The capital letters are the names of routers. An arbitrary metric with value between 1 and 3 is used. The initial metrics of all the links are indicated in the parentheses. The incremental functions $fp(c)=4c$ and $fe(c)=2c$ are used in this example. Two paths from router S to router D are demanded.

On the first run of the Dijkstra algorithm, the shortest path S->A->D with metric 3 is obtained.

The incremental function fp is applied to increase the metric of the link S-A and A-D. fe is applied to increase the metric of the link A-B and A-C. Figure 3 shows the link metrics after the increment.

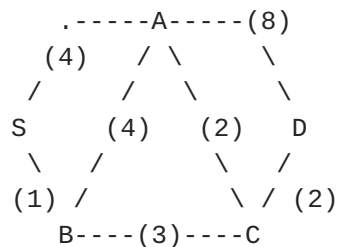


Figure 3

On the second run of the Dijkstra algorithm, the second path S->B->C->D with metric 6 is obtained.

As mentioned in [Section 8.5](#), the Multipath Dijkstra Algorithm does not guarantee strict disjoint paths in order to avoid choosing inferior paths. For example, given the topology in Figure 4, two paths from node S to D are desired. On the top of the figure, there is a high cost path between S and D.

If a algorithm tries to obtain strict disjoint paths, the two paths obtained will be S--B--D and S--(high cost path)--D, which are extremely unbalanced. It is undesirable because it will cause huge delay variance between the paths. By using the Multipath Dijkstra

algorithm, which is based on the punishing scheme, S--B--D and S--B--C--D will be obtained.

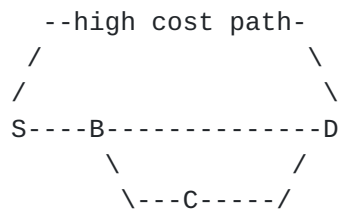


Figure 4

Authors' Addresses

Jiazi Yi
Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33 (0) 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) 2 40 68 30 50
Email: Benoit.Parrein@polytech.univ-nantes.fr
URI: <http://www.irccyn.ec-nantes.fr/~parrein>

