Network Working Group
Internet-Draft
Expires: December 25, 2006

K. Weniger
Panasonic
K. Mase
Niigata University
June 23, 2006

PDAD-OLSR: Passive Duplicate Address Detection for OLSR
draft-weniger-autoconf-pdad-olsr-01

Status of this Memo

Copyright Notice

Abstract

This draft proposes PDAD-OLSR, a solution for configured address uniqueness maintenance in MANETs running the OLSR protocol. It utilizes the Passive Duplicate Address Detection (PDAD) concept, which enables nodes to passively detect duplicate addresses in the network (e.g., occurring after network merging) by analyzing received routing protocol messages. Due to its passive nature, PDAD-OLSR is very efficient in terms of bandwidth consumption. Moreover, it can

prevent the contamination of routing tables with wrong routing information resulting from address conflicts.

Table of Contents

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

The terminology of OLSR [2] is used.

Conflicting address: An address that is not unique in the network, i.e., two MANET network interfaces in the same MANET are configured with the same address.

Conflicting nodes: Two or more nodes configured with the same (conflicting) address.

2. Introduction and Motivation

Address autoconfiguration is a fundamental issue in MANETs, since routing protocols assume that the involved network interfaces are configured with unique IP addresses and manual assignment is often not applicable. Solutions for traditional IP networks such as DHCP [5], DHCPv6 [6], zeroconf [7] or IPv6 Stateless Address Autoconfiguration [8] cannot be applied to MANETs due to their special properties such as the dynamic multi-hop nature. See [3] and [18] for an overview of problems and approaches for MANET address autoconfiguation.

Duplicate Address Detection (DAD) is an essential part of address autoconfiguration, especially for stateless protocols. In MANETs, addresses may become duplicate when they are already assigned to nodes, e.g., after optimistic address assignment or after two or more independently configured MANETs merge to one network. This problem is also referred to as configured address uniqueness maintenance problem. Not re-using addresses in different (unconnected) MANETs by constructing MANET-local IP addresses based on pre-configured globally unique IDs such as IEEE MAC addresses may seem to solve this problem without a DAD mechanism, however, for the following reasons we think that this is not a general solution:

- Addresses based on pre-configured globally unique IDs are usually not 100\% globally unique, e.g., a IEEE MAC address (or the IP address itself) configured at a specific network interface may be changed by the user, devices with duplicate MAC addresses exist on the market (non-registered or erroneous manufacturing), and some devices may not have globally unique IDs (e.g., sensors)

- Addresses based on pre-configured globally unique IDs may have negative implications on privacy [4]

- Since this approach requires long addresses to allow the addressing of all existing devices, it is only possible with IPv6, not with IPv4

- Long addresses result in a significant increase of signaling traffic, e.g., of the routing protocol. Dynamically assigning locally unique addresses and re-using them in different (unconnected) MANETs is more efficient, since such addresses may be shorter or can easily be compressed to shorter addresses in routing protocol packets. [17] first proposed the compression of addresses in routing protocol messages. OLSRv2 [14] and DYMO [15] support a special form of address compression.

In case (large) parts of the address are generated from random

numbers (e.g., as proposed in [10]), issue 1 and 2 may not be a problem anymore, but issue 3 and 4 remain. Because of these issues and because different nodes in a network can use different methods for address generation, we think that a DAD solution for the configured address uniqueness maintenance problem is needed. However, the solution should be very bandwidth-efficient to justify its use in cases where the probability of a conflict is very low.

This draft proposes a very bandwidth-efficient solution for configured address uniqueness maintenance in a network running the OLSR protocol [2]. The solution is suitable for any kind of addresses exchanged in routing protocol messages, be it MANET-local or globally routable addresses of IPv4 or IPv6. It supports both MANET single- and multi-subnet architecture (see Section 4.12) and is in line with the proposed framework for autoconfiguration [13]. The generation and assignment of addresses is out of scope of this draft. Multiple network interfaces and OLSRv2 are not considered in this version of the draft.

3. Overview of the Passve DAD Concept

The PDAD concept for MANETs was first proposed in [19] and later refined, analyzed, and integrated in a complete address autoconfiguration solution [17]. The initial solution was modular to support multiple routing protocols, namely OLSR, AODV, and FSR. Later, multiple drafts proposed the use of PDAD in the IETF specifically for OLSR [12] [11] [16].

PDAD defines a set of rather simple algorithms that allows nodes to detect address conflicts in the network based on routing protocol anomalies. A specific combination of algorithms is supposed to to detect all conflicts in the network running a specific routing protocol. The basic idea of PDAD is to exploit the fact that some protocol events occur in case of a duplicate address, but (almost) never in case of a unique address.

PDAD-enabled nodes derive information about the state of the routing protocol daemon running on another node's network interface and configured with a specific address (e.g., address A) from incoming routing protocol messages. If the receiver's address equals A, the information related to address A in the incoming message is compared to information associated with the state of the receiver's routing protocol daemon in order to detect a possible conflict of address A. If the receiver's address does not equal A, it compares the information from the message to information associated to the state derived from another recently received routing protocol message containing information about address (address A), i.e., PDAD allows the detection of conflicts by intermediate nodes that have unique addresses.

Since the state of a routing protocol daemon is changing over time, a node receiving a routing protocol message must have information about the time this routing protocol message has been sent. Without synchronized clocks and additional information in the messages, this time can only be estimated. Here, it is assumed that the time interval during which a specific routing protocol message is completely disseminated in the network can be estimated to be less than a time span T_D. In this case, routing protocol messages received by a node can never be older than T_D. Note that "complete dissemination" of a message does not mean that the message reaches all nodes, it just means that it is not forwarded anymore in the network. T_D can be quite large (e.g., in the order of minutes) and can be different for different routing protocol messages depending on the maximum number of times a message is fowarded, e.g., for HELLO and TC messages. The stated assumption usually holds quit well in reality, since routing protocols use a duplicate cache, nodes do not store to-be-forwarded routing information for unlimited time and

queuing and media access delays are usually somehow bounded. In fact, all well-known ad hoc routing protocols implicitly assume a certain maximum dissemination time T_D, since otherwise they would not be able to distinguish fresh from stale routing information after sequence number wrap-arounds. In case the estimate for the time span T_D is wrong, PDAD may generate false alarms and nodes with unique address may be forced to change their address.

4. PDAD Algorithms for OLSR

In the following, PDAD algorithms for OLSR [2] are presented. The algorithms were first proposed in [17] and specify what a node has to do to detect duplicate addresses based on incoming routing protocol messages. The algorithms utilize different parameters in TC and HELLO messages such as link states (i.e., neighbor interface addresses), link codes, (message) sequence numbers, and addresses in OLSR routing protocol messages as well as addresses in the IP header. For better undestanding, the algorithms are illustrated in examplary scenarios in Appendix A. A node must implement all of the algorithms to be able to detect all conflicts in the network in all possible scenarios. PDAD-OLSR considers that the MPR selection may be affected by duplicate addresses in the network, which may result in a limited dissemination of routing protocol messages in the network (see Appendix A.10).

4.1. Data Structures and Parameters

In addition to the OLSR data structures (or information repositories), each node conceptually maintains two tables for PDAD: a Last Received Protocol Messages (LRM) table and a Neighbor History (NH) table.

The Last Received Protocol Messages (LRM) table contains information about the last TC and HELLO protocol message received from a specific originator address. It has the following structure:

- Originator Address
- Message Type
- Message Sequence Number
- Neighbor Interface Addresses (with corresponding Link Codes if HELLO message)
- Receive Time

The Neighbor History (NH) table contains the history of neighboring node addresses and is build from received HELLO messages. Entries older than T_D can be deleted. The entries have the following structure:

- Neighbor Interface Address
- Last time the receiver has selected this Neighbor Interface Address as MPR

- Last time the receiver has been selected as MPR by this Neighbor Interface Address

- Receive Time of HELLO message

The following protocol parameters are used:

Parameter name	Meaning	Default value
SN_MAX	Maximum message sequence number value	65535
T_D (TC)	Maximum dissemination time of TC messages in the network	30s
T_D (HELLO)	Maximum dissemination time of HELLO messages in the network	2s
SN_RATE	Maximum rate of message sequence number incrementation per node	5/s

4.2. PDAD-Source Address (SA)

If a node receives a TC or HELLO message, it compares the source address in the IP header to its own address. If they equal, a conflict of this address is detected. If the message is a HELLO messages, the originator address can be used instead of the source address in the IP header.

The rationale behind this algorithm is that the IP source address is always the address of the last forwarder. This is true for OLSR, since it uses application-layer forwarding of TC messages. Note that an originator address in a TC message which is equal to the receiver's address is not an indication for an address conflict, since a node can receive TC messages originated by itself and forwarded by neighboring nodes.

Care must be taken when implementing PDAD-SA, since broadcast messages must not be duplicated within the sending node and internally delivered to the IP stack as received message. This would generate false alarms.

The algorithm works completely passive and can detect conflicts between neighboring nodes only.

4.3. PDAD-Sequence Numbers (SN)

If a node receives a TC or HELLO message, it compares the originator address with its own address. If they equal and the sequence number in the message is higher than the receiver's sequence number, a conflict of the originator address is detected. However, false alarms can occur in case of sequence number wrap-arounds. Hence, a conflict must not be assumed if a wrap-around may be the reason for the sequence number inconsistency. A mechanism to detect a possible sequence number wrap-around is described in section Section 4.11.

The rationale behind this algorithm are that a node receiving its own TC messages forwarded by other nodes cannot have a sequence number large than the node's internal sequence number counter. It is assumed that no wrap-around has occurred, that sequence numbers are incremented with a certain maximum rate and that each node only increments its own internal sequence number counter.

The algorithm works completely passive and can detect conflicts between nodes that are any number of hops away from each other.

4.4. PDAD-Sequence Number Difference (SND)

If a node receives a TC or HELLO message, it compares the sequence number in the message with the sequence number in the previously received message from the same originator address and with the same message type. If the sequence number difference is higher than a threshold SNDTHRES, a conflict of the originator address is detected. SNDTHRES can be computed as follows: SNDTHRES=(|T_R1-T_R2|+T_D)*SN_RATE with T_R1 and T_R2 the receive time of message 1 and 2, respectively. Note that the computation of the sequence number difference must consider wrap-arounds, e.g., by calculating the difference with min(|SN1-SN2|,SN_MAX-|SN1-SN2|).

The rationale behind this algorithm is that there is a relation between the time between a node has originated two TC messages and the sequence number in those TC messages. Therefore, it is assumed that sequence numbers are incremented with a certain maximum rate and that each node only increments its own internal sequence number counter.

The algorithm works completely passive and can detect conflicts between nodes that are any number of hops away from each other.

4.5. PDAD-Sequence Numbers Equal (SNE)

If an intermediate node receives a TC or HELLO message, it searches its LRM table for a message with the same message type value and the

same tuple <sequence number, originator address>. If a matching entry is found, the node compares the neighbor interface addresses in both messages. A conflict is detected if they differ. Only messages that have arrived within the preceding time span SN_MAX/SN_RATE-T_D should be considered due to possible sequence number wrap-arounds.

The rationale behind this algorithm is that the tuple <sequence number, originator address> uniquely identifies a messages originated by a specific node.

The algorithm works completely passive and can detect conflicts between nodes that are any number of hops away from each other.

4.6. PDAD-SNs Always Increment (SNI)

If a node receives a HELLO message, it compares the sequence number in the message with the sequence number in the previous HELLO message from the same originator address. If the sequence number in the new message is lower, a conflict of the originator address is detected. Again, this is only true if a sequence number wrap-around cannot be the reason for the inconsistency. A mechanism to detect a possible sequence number wrap-arounds is described in section Section 4.11.

The rationale is that HELLO messages sent by a specific node are received in the order they are sent (i.e., with increasing sequence numbers), since they are not forwarded and hence cannot "overtake" each other.

The algorithm works completely passive and can only detect conflicts between nodes that are at most two hops away from each other.

4.7. PDAD-Neighborhood History (NH)

If a node receives a TC message, it checks whether its own address is part of the neighbor interface addresses in the TC message. If this is the case and the link code indicates a bi-directional link, the node searches the originator address in its NH table. If it cannot find the address in the table with a receive time higher than the current time minus T_D, a conflict of the node's address is detected. Otherwise, the node additionally checks whether the NH table indicates that the node has selected the found address as MPR within the last T_D. If this is not the case, a conflict is detected.

The rationale behind this algorithm is that a TC message only contains neighbors that have selected the originator address as MPRs and that this requires a bi-directional link. If all addresses in the network are unique, a node having an address equal to one of the neighbor interface addresses in a received TC message must have been

a neighbor of the originator address.

The algorithm works completely passive and can detect conflicts between nodes that are any number of hops away from each other.

4.8. PDAD-Link States (LS)

If a node receives a TC message with its own address as originator address, it searches in its NH table for each of the neighbor interface addresses. If at least one address is not found with a receive time higher than the current time minus T_D, a conflict of the originator address is detected. If all addresses have been found, but the NH table indicates that the node's address has not been selected as MPR by the found address within the last T_D, a conflict is detected.

The rationale is that if the message has been originated by the receiver, it must only contain addresses of recent neighbor interfaces.

The algorithm works completely passive and can detect conflicts between nodes that are any number of hops away from each other.

4.9. PDAD-extended Neighborhood History (eNH)

If a node receives a TC message, it checks for each neighbor interface address in the message if it is a neighbor. This can be done by checking the OLSR neighborhood or local link information base or the LRM table. For each found neighbor address, the node searches in the LRM table for previously received HELLO message from this address. For each found HELLO message (not older than T_D), it checks whether the originator address of the TC message is contained in the set of neighbor interface addresses of the found HELLO message. If this is not the case, this is a hint for a conflict of the originator address of the HELLO message. If this is the case, the node additionally checks the link codes of the respective neighbor interface addresses in the HELLO message. If they indicate that the originator address of the TC message has not been selected as MPR within the last T_D by the originator address of the HELLO message, this is another hint for a conflict of the originator address of the HELLO message. However, the receiver cannot be sure whether a conflict exists or not, since it is not aware of the complete neighbor history of the respective neighbor. Hence, the receiver forwards the TC message even if it is not selected as MPR and would normally not forward this TC message. The originator of the HELLO message is then able to detect the conflict by applying the PDAD-NH algorithm on the TC message. Note that the forwarded TC message must be marked as "PDAD eNH hint message" (e.g., by a flag)

and that PDAD-eNH may not be applied to such TC messages. Otherwise PDAD-eNH on other nodes may suspect a conflict of the address of the hint TC message sender. How the marking is exactly done is TBD.

This algorithm is basically the PDAD-NH algorithm executed on behalf of a neighboring node. Minimal additional signaling is needed, which means that this algorithm is not completely passive. A possible optimization to reduce the additional signaling would be to store the neighborhood history of neighbors in each node as learned from received HELLO messages. However, this would require extra amount of memory.

The algorithm works semi-passive and can detect conflicts between nodes that are any number of hops away from each other.

4.10. Summary

This section summarizes the properties of the PDAD algorithms.

Algorithm	Relevant parameters in message	Potentially conflicting nodes	Maximum distance between conflicting nodes	Completely passive
PDAD-SA	sequence number, IP source address	originator/forwarder and receiver	1 hop	yes
PDAD-SN	sequence number, originator address	originator and receiver	n hops	yes
PDAD-SND	sequence number, originator address	both originators	n hops	yes
PDAD-SNE	sequence number, originator address	both originators	n hops	yes

PDAD-SN I	sequence number, originator address	both originators	2 hops	yes	
PDAD-LS	neighbor addresses, originator address	originator and receiver	n hops	yes	
PDAD-NH	neighbor addresses, originator address	neighbor of originator and receiver	n hops	yes	
PDAD-eNH	neighbor addresses, originator address	neighbor of originator and neighbor	n hops	no	

4.11. Detecting Sequence Number Wrap-arounds

Wrap-arounds occur when the sequence number value reaches SN_MAX and, after another incrementation, starts again from 0. Wrap-around events are rare if SN_MAX is high and the sequence numbers are initialized to a low value. If only unique addresses exist in the network and a message dissemination time of T_D as well as a maximum sequence number increase rate of SN_RATE is assumed, the maximum difference of the sequence number value from receiver and sender point of view is SN_THRES=SN_RATE*T_D. Consequently, a wrap-around can only be the reason for a sequence number inconsistency, if the lower sequence number value s1 is smaller than SN_THRES and the higher sequence number value s2 is bigger than SN_MAX-SN_THRES+s1.

4.12. Support for Multi-Subnet MANET Architecture

The descriptions in this document assumes a single-subnet MANET architecture, but a multi-subnet MANET architecture as proposed in [9] is supported as well. According to the multi-subnet architecture, all MANET routers are assumed to configure their MANET router's OLSR interfaces (which is a loopback interface) with a unique subnet prefix. Assuming that the interface is configured with an address from this prefix, the mechanisms in this document can be used to ensure the uniqueness of the subnet prefix in the network by

additionally masking the host part of the address. In case all OLSR interfaces use the same host part, the masking is not necessary.

5. Conflict Resolution and Related Issues

5.1. Conflict Resolution

If an algorithm detects a conflict of the receivers's address, the node changes its IP address in order to resolve the conflict. If a conflict is detected by an intermediate node, the conflicting nodes must somehow be informed about the conflict so that they are able to change their address. This document described two options for conflict notification.

5.1.1. Option A

This option requires sending a dedicated message via unicast to the duplicate address. The format for this message is specified in Section 6. The destination address is set to the conflicting address. The very conflicting node that caused the inconsistent routing information in the message triggering the conflict detection should change its address. To achieve that, the message should be sent towards the source address in the IP header of the received routing protocol message that triggered the conflict detection. This node then uses its routing table as usual to forward the message to the correct conflicting node. Controlling the next hop towards the conflicting address can be done by using IPv4 loose source routing, IPv6 routing header, or IPv4/IPv6 tunneling. How this is exactly done is TBD.

5.1.2. Option B

This option informs all nodes in the network about a detected address conflict by adding conflicting addresses to TC and HELLO messages or marking duplicate addresses as such. It was first proposed in [11]. How the marking is exactly done is TBD.

5.2. Preventing Routing Table Contamination

To prevent the contamination of the routing table with false routing information emerging from an address conflict, information about duplicate addresses MUST NOT be used for calculating routes. If option A is used, a TC message that was used to detect the conflict must be ignored for routing table calculation or must not be forwarded, so that the routing tables in other nodes are not contaminated. If option B is used, TC messages are forwarded, but addresses marked as duplicate in the message MUST be ignored for routing table calculation.

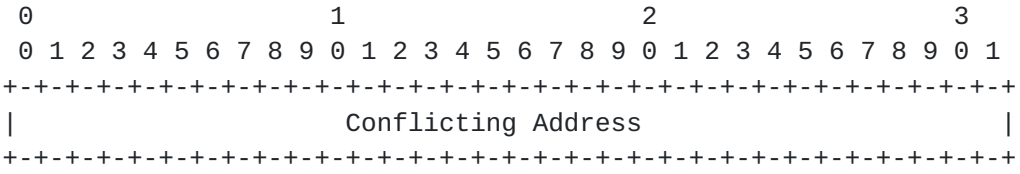
5.3. Handling Address Changes

Care must be taken when a node changes its IP address, because the bidirectional link states and the MPR selection states of the OLSR protocol daemon are based on the context of the old address. Hence, a node has to set all its link states to asymmetric and remove all addresses from its MPR selector set. Without these modifications, PDAD-NH would immediately detect a conflict of the new address even if it is unique.

6. Message Formats

6.1. Conflict Resolution Message

The message is encapsulated in an OLSR packet header. The message only contains the conflicting address. The message is send to the conflicting address over the last forwarder of the very routing protocol message that triggered the conflict detection. This can be done by IP tunneling, IPv6 routing header or IPv4 loose source option. Message type is TBD.



6.2. Changes to TC and HELLO Message

How duplicate addresses are added and marked in TC and HELLO messages is TBD.

7. Security Considerations

TBD

8. References

8.1. Normative References

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

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

[3] Singh, S., "Address autoconfiguration for MANETs: definition and problem statement", draft-singh-autoconf-adp-03 (work in progress), March 2006.

8.2. Informative References

[4] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[5] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[6] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[7] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, March 2005.

[8] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[9] Thaler, D., "Multi-Subnet MANETs", draft-thaler-autoconf-multisubnet-manets-00 (work in progress), February 2006.

[10] Jelger, C., "MANET Local IPv6 Addresses", draft-jelger-autoconf-mla-00 (work in progress), April 2006.

[11] Mase, K. and C. Adjih, "No Overhead Autoconfiguration OLSR", draft-mase-manet-autoconf-noaolsr-01 (work in progress), April 2006.

[12] Baccelli, E., "OLSR Passive Duplicate Address Detection", draft-clausen-olsr-passive-dad-00 (work in progress), July 2005.

[13] Mase, K., "A common framework for autoconfiguration of stand-

alone ad hoc networks", draft-mase-autoconf-framework-01 (work in progress), February 2006.

[14] Clausen, T., "The Optimized Link-State Routing Protocol version 2", draft-clausen-manet-olsrv2-01 (work in progress), September 2005.

[15] Perkins, C. and I. Chakeres, "Dynamic MANET On-demand (DYMO) Routing", draft-ietf-manet-dymo-04 (work in progress), March 2006.

[16] Weniger, K., "PDAD-OLSR: Passive Duplicate Address Detection for OLSR", draft-weniger-autoconf-pdad-olsr-00 (work in progress), February 2006.

[17] Weniger, K., "PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks", IEEE Journal of Selected Areas of Communications (JSAC), Vol. 23 No. 3, March 2005.

[18] Weniger, K., "Address Autoconfiguration in Mobile Ad Hoc Networks: Current Approaches and Future Directions", IEEE Network Magazine , July 2004.

[19] Weniger, K., "Passive Duplicate Address Detection in Mobile Ad hoc Networks", IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, USA, March 2003.

Appendix A. Illustration of PDAD Algorithms

A.1. Notation

Node "A" and its OLSR protocol daemon states are depicted with "A(address,sequence number,{neighbor interface address 1, neighbor interface address 2,...})". Non-relevant values as well as broadcast and multicast addresses are represented by "*". Neighboring nodes are connected with dashes (e.g., "A()----B()"), nodes that are not necessarily neighbors with dashes and points (e.g., "A()-...-B()"). The notation for addresses in the IP header is "[src->dst]". TC messages are depicted with "TC(originator address, message sequence number, {neighbor interface address 1, neighbor interface address 1,...})" (HELLO messages accordingly with "HE(..)"). "#(X)" denotes that the node has detected a conflict of address X.

A.2. PDAD-SA

An example scenario is given in Figure 1. Node A is configured with address 1 and sends a TC (or HELLO) message. Node B is a neighbor of node A and is configured with the same address 1. Upon receiving the message and comparing the IP source address with its own address, it detects a conflict of its own address.

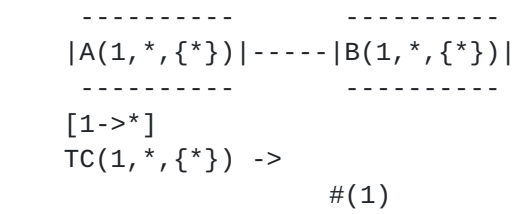


Figure 1: Example of PDAD-SA

A.3. PDAD-SN

An example scenario is given in Figure 2. Node A with address 1 sends a TC message. Node B is located somewhere in the network and is configured with the same address 1. Upon receiving the TC message, node B compares originator address and sequence number with its own address and sequence number counter. Since the addresses are equal and the sequence number in the message is higher than its own counter, it detects the conflict of its own address (it is assumed that node B has checked that a wrap-around cannot be the reason for the sequence number inconsistency).

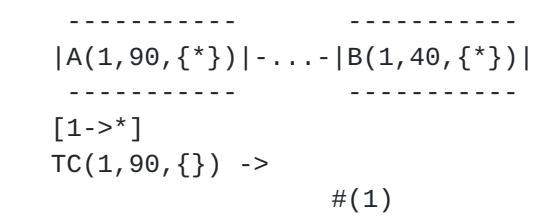


Figure 2: Example of PDAD-SN

A.4. PDAD-SND

An example scenario is given in Figure 3. Node A with address 1 sends a TC message. Its message sequence number counter value is 5. Node C is configured with the same address 1, but its message sequence number counter value is 2000. After receiving the TC message sent by node A, node B stores the information contained in the message. When the TC message sent by node C is received by node B, it compares the sequence number with the sequence number of the last TC message received from the same originator address. Assuming a threshold SDNTHRES lower than 1995, it detects a conflict of address 1.

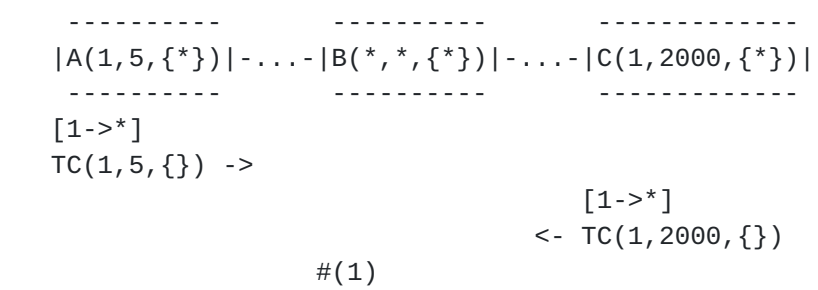


Figure 3: Example of PDAD-SND

A.5. PDAD-SNE

An example scenario is given in Figure 4. Node A with address 1 sends a TC message. Its message sequence number counter value is 5 and the neighbor interface addresses are 3 and 4. Node C is configured with the same address 1 and has the same message sequence number counter value, but the neighbor interface addresses are 6 and 7. After receiving the TC message sent by node A, node B stores the information in the message. When the TC message sent by node C is received by node B, it compares the sequence number with the sequence number of the last TC message received from the same originator

address. Since they are equal, it compares the neighbor interface addresses. Node B detects a conflict of address 1, because at least one neighbor interface address is different.

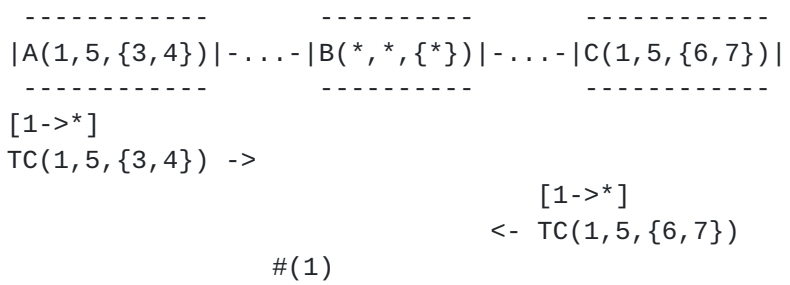


Figure 4: Example of PDAD-SNE

A.6. PDAD-SNI

An example scenario is given in Figure 5. Node A with address 1 has a message sequence number counter value of 5. Node B is a neighbor of node A and node C is a neighbor of node B. Node C is also configured with address 1. Its message sequence number counter value is 4. After receiving the HELLO message sent by node A, node B stores the information contained in the message. After the HELLO message sent by node C is received by node B, node B compares the sequence number with the sequence number in the last HELLO message received from the same originator address. Since the second HELLO message has a lower sequence number, node B detects a conflict of address 1 (it is assumed that node B has checked that a wrap-around cannot be the reason for the sequence number inconsistency).

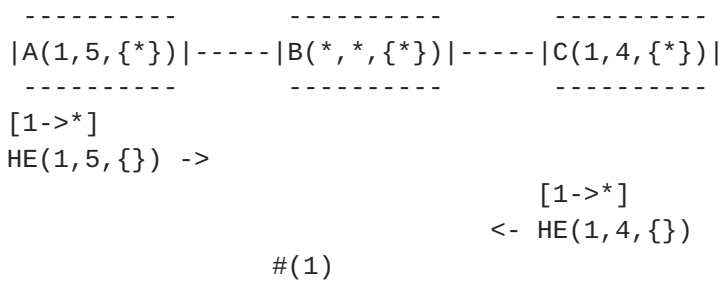


Figure 5: Example of PDAD-SNI

A.7. PDAD-NH

An example scenario is given in Figure 6. Node A has address 1 and a Neighbor History (NH) cache containing the addresses 4 and 5. Node B is located somewhere in the network and is configured with address 2. A Node C is a neighbor of node B and is also configured with address 1. Node B sends a TC message. Upon receiving the message, node A notices that its own address is part of the neighbor interface addresses of the TC message. Hence, it checks whether the originator address 2 has recently been a symmetric neighbor by consulting its NH table. Since the check is negative, a conflict of address 1 is detected.

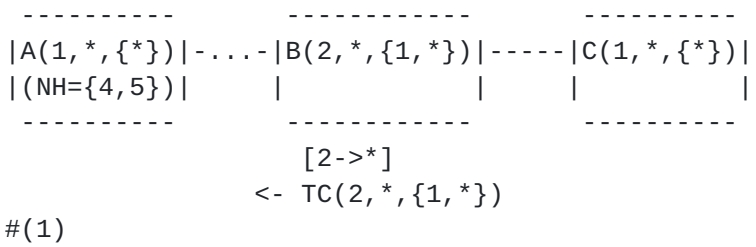


Figure 6: Example of PDAD-NH

A.8. PDAD-LS

An example scenario is given in Figure 7. Node A has address 1 and a Neighbor History (NH) cache containing the addresses 4 and 5. Node B is located somewhere in the network, is also configured with address 1, and sends a TC message. Upon receiving the message, node A notices that its own address is equal to the originator address. Hence, it checks whether at least one of the neighbor interface addresses has not recently been a neighbor by consulting its NH table. Since this is the case, a conflict of address 1 is detected.

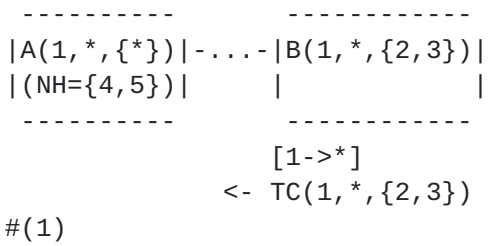


Figure 7: Example of PDAD-LS

A.9. PDAD-eNH

An example scenario is given in Figure 8. Node A has address 1 and a Neighbor History (NH) cache containing the addresses 2 and 5. Node B is a neighbor of node A and is configured with address 2. Node C is located somethere in the network and has the address 3. Node D is neighbor of node C and is also configured with address 1. After node A has sent a HELLO message, Node C sends a TC message. Upon receiving the messages, node B notices that a neighbor interface address in the TC message is equal to the address of one of its neighbors (1). It then checks the neighbor interface addresses of this neighbor (1) as learned from the last HELLO message from this address and notices that the originator address of the TC message (3) is not part of this set (2). Hence, it concludes that an address conflict may exist. It marks and forwards the TC message even if it is not elected as MPR node for this TC message. Node A receives the TC message and detects the conflict after applying the PDAD-NH algorithm (since address 3 is not part of node A's NH table).

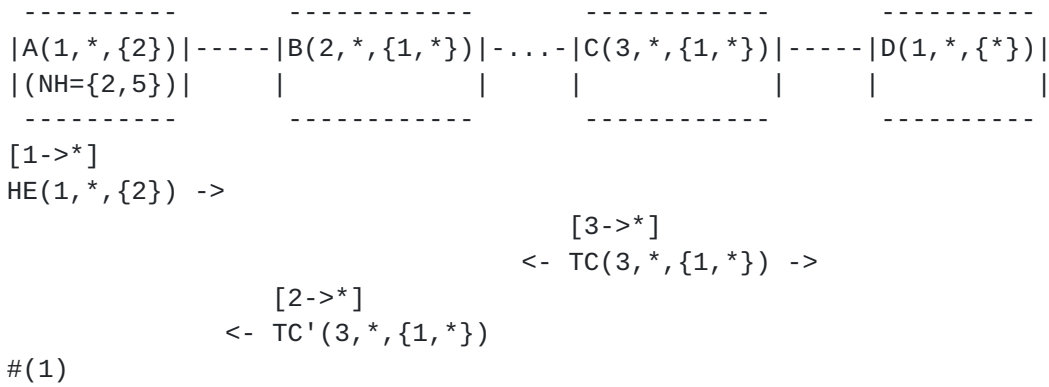


Figure 8: Example of PDAD-eNH

A.10. Effects of Address Conflicts on MPR Selection

Address conflicts within 4 hops distance may influence MPR selection and may lead to limited dissemination of TC messages. For example, in the scenario shown in Figure 9, node C would not forward TC messages received by node D and vice versa, since both nodes assume that they don't have 2-hop-neighbors (only a 1-hop-neighbor with address 2). The network is virtually partitioned with respect to TC message dissemination. This may be a problem for PDAD algorithms based on TC messages, for instance, PDAD-NH or PDAD-SN cannot detect the conflict of address 1 in the example scenario. However, all conflicts within 4 hops can be detected with the combination of algorithms proposed in this draft. After resolving these conflicts,

TC messages are again disseminated correctly and conflicts with more than 4 hops between the conflicting nodes can be detected and resolved. In the example scenario, the conflict of address 2 between node B and node E can be detected by PDAD-eNH. After this conflict has been resolved, the conflict of address 1 between node A and F can be detected, e.g., by PDAD-SN or PDAD-NH. See [17] for more details about this problem.

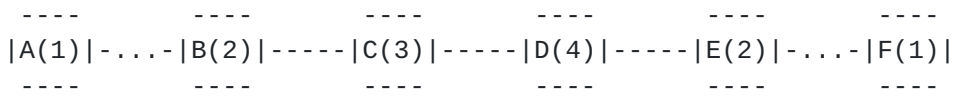


Figure 9: Example of MPR selection influenced by address conflict

Authors' Addresses

Kilian A. Weniger
Panasonic R&D Center Germany
Monzastr. 4c
Langen 63225
Germany

Phone: +49 6103 766 137
Email: kilian.weniger@eu.panasonic.com

Kenichi Mase
Niigata University
Ikarashi
Niigata-shi 950-2181
Japan

Phone: +81 25 262 7446
Email: mase@ie.niigata-u.ac.jp

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.