DNA WG
Internet-Draft
Expires: July 21, 2006

JH. Choi
Samsung AIT
E. Nordmark
SUN Microsystems
January 17, 2006

DNA with unmodified routers: Prefix list based approach
draft-ietf-dna-cpl-02.txt

Status of this Memo

Copyright Notice

Abstract

Upon establishing a new link-layer connection, a host determines whether a link change has occurred, that is, whether or not it has moved at layer 3 and therefore needs new IP configuration. This draft presents a way to robustly check for link change without assuming any changes to the routers. We choose to uniquely identify each link by the set of prefixes assigned to it. We propose that, at each attached link, the host generates the Complete Prefix List, that

is, a prefix list containing all the valid prefixes on the link, and when it receives a hint that indicates a possible link change, it detects the identity of the currently attached link by consulting the existing prefix list. This memo describes how to generate the Complete Prefix List and to robustly detect the link identity even in the presence of packet loss.

Table of Contents

1. Introduction

When a host establishes a new link-layer connection, it may or may not have a valid IP configuration, such as the subnet prefixes or the default router addresses, for the link. Though the host has changed its network Point of Attachment (at layer 2), it may still be at the same link (at layer 3). The term 'link' used in this document is as defined in RFC 2461 [1], which is a layer 3 definition. NOTE that that definition is completely different from the definition of the term 'link' in IEEE 802 standards.

Thus the host needs to check for a link change, i.e. it needs to verify whether it is attached to the same or a different link as before [4]. The host can keep current IP configuration if and only if it remains at the same link.

A host receives the link information from RA (Router Advertisement) messages. However, as described in 2.2. [4], it's difficult for a host to correctly detect the identity of a link with a single RA. None of the information in an RA can indicate a link change properly. Neither router address nor prefixes will do.

It may be better to design a new way to represent the identity of a link, and/or add new pieces of information to RA or RS (Router Solicitation) messages. Several new approaches to properly indicate link change have been considered by the design team - see [10].

However, even if some such new scheme is standardized and implemented, hosts would still need to cope with routers which do not (yet) implement such a scheme. Thus it makes sense to write down the rules for how to robustly detect the link identity without assuming any changes to the routers, which is the purpose of this document.

2. Prefix list based approach

2.1 Approach

Currently there is one thing which can represent the identify of a link,

'The set of all the valid and global prefixes assigned to a link.'

If a host has the complete list of all the assigned prefixes, it can properly determine whether a link change has occurred. If the host receives an RA containing one or more prefixes and none of the prefixes in it matches the previously known prefixes for the link, then it is assumed to be a new link.

This works because each and every valid global prefix on a link must not be used on any other link thus the sets of global prefixes on different links must be disjoint [3].

This is the case even as there is renumbering. During graceful renumbering a prefix would gradually have its (preferred and valid) lifetimes decrement, until the valid lifetime reaches zero. Some point after the valid lifetime has reached zero, the prefix may be reassigned to some different link. Even during 'flash' renumbering, when the prefix isn't allowed to gracefully move through the deprecated state [2], independently of DNA, the prefix needs to be advertised with a zero valid lifetime on the old link before it can be reassigned. Thus we can assume that a prefix with a non-zero valid lifetime can at most be assigned to one link at any given time.

For the purposes of determining the prefixes, this specification uses both 'on-link' and 'addrconf' prefixes [1], that is, prefixes that have either the 'on-link' flag set, the 'autonomous address-autoconfiguration' flag set, or both flags set. This is a safe approach since both the set of valid on-link and the set of valid addrconf prefixes must be uniquely assigned to one link.

While the approach is conceptually simple, the difficulty lies both in ensuring that the host knows the Complete Prefix List for a single link, and preventing prefixes from possibly different links to be viewed as the prefixes for a single link. This is challenging for several reasons: A single RA is not required to include all prefixes for the link, RAs might be subject to packet loss, new routers and new prefixes (due to renumbering) might appear at any time on a link, and the host might move to a different link at any time.

If the prefix list determination is incorrect, there can be two different types of failures. One is detecting a new link when in

fact the host remains attached to the same link. The other is failing to detect when the host attaches to a different link. The former failure is undesirable because it might trigger other protocols, such as Mobile IPv6 [5], to do unneeded signaling, thus it is important to minimize this type of failure. The latter type of failure can lead to long outages when the host is not able to communicate at all, thus these failures must be prevented.

2.2 Assumptions

In this approach, we assume that an interface of a host can not be attached to multiple links at the same time. Though this kind of multiple attachments is allowed in neither Ethernet nor 802.11b, it may be possible in some Cellular System, especially CDMA.

This assumption implies that, should the host use a layer 2 technology which can be multiply connected, this needs to be represented to the DNA (and layer 3 on the host in general), as separate (virtual) interfaces, so that the DNA module can associate each received RA message with a particular (virtual) interface.

We also assume that when a host changes its Point of Attachment, the DNA module will be notified of the event using some form of 'link UP' event notification, and that the DNA module determines which RAs arrived before the event and which arrived after the event [9]. This assumption places some requirements on the host implementation, but does not place any assumptions on the layer 2 protocol.

It is possible to have CPL operate in less robust fashion when the implementation does not provide such a 'link UP' event notification. We mention this possibility in Section 5.

2.3 Overview

Hints are used to tell a host that a link change might have happened. This hint itself doesn't confirm a link change, but can be used to initiate the appropriate procedures [4].

In order to never view two different links as one, it is critical that when the host might have attached to a link, there has to be some form of hint. This hint doesn't imply that a movement to a different link has occurred, but instead, in the absence of such a hint there could not have been an attachment to a different link.

If the IP stack is notified by the link layer when a new attachment is established (e.g., when associating to a different access point in 802.11), this will serve as such a hint. It helps to reduce the risk that the assignment of an additional prefix to a link will be

misinterpreted as being attached to a different link. Note that this hint is merely a local notification and does not require any protocol changes. For instance, in many implementations this would be a notification passed from a link-layer device driver to the IP layer [9].

Once a hint is received the host will start to collect a new set of valid prefixes for the possibly different link, and compare them with the valid prefixes known from before the hint. If there is one or more common prefixes it is safe to assume that the host is attached to the same link, in which case the prefixes learned after the hint can be merged with the prefixes learned before the hint. But if the sets of valid prefixes are disjoint, then at some point in time the host will decide that it is attached to a different link.

The process of collecting valid prefixes starts when the host is powered on and first attaches to a link.

Since each RA message isn't guaranteed to contain all valid prefixes it is a challenge for a host to attain and retain the Complete Prefix List, especially when packets can be lost on the link.

The host has to rely on approximate knowledge of the prefix list using RS/ RA exchanges. Just as specified in [1], when the host attaches to a potentially new link, it sends an RS (Router Solicitation) message to All-Router multicast address, then waits for the solicited RAs. If there was no packet loss, the host would receive the RAs from all the routers on the link in a few seconds thereby knowing all the valid prefixes on the link. Taking into account packet loss, the host may need to perform RS/ RA exchanges multiple times to corroborate the result.

When a hint indicating a possible link change happens, if the host is reasonably sure that its prefix list is complete, it can determine whether it is attached to the same link on the reception of just one RA containing one or more valid prefixes.

Otherwise, to make matters certain, the host may need to attempt further procedures. A first step to clarify link identity is to wait for all RAs which would have been sent in response to the RS. A further step is to send multiple RSs (and wait for the resulting RAs).

All tracking of the prefix lists must take the valid lifetime of the prefixes into account. The prefix list is maintained separately per network interface.

3. DNA based on the Complete Prefix List

We choose to identify a link by the set of valid prefixes that are assigned to the link, and we denote this 'the Complete Prefix List'. Each link has its unique Complete Prefix List. We also say that the prefix list is complete if all the prefixes on the link belong to it.

In case that a host has the Complete Prefix List, it can properly determine whether it is attached to the same link or not, when it receives a single RA message after a hint of possible link change.

This section presents a procedure to generate the Complete Prefix List and a way to detect the link identity based on the existing prefix list even in the presence of packet losses.

3.1 Complete Prefix List generation

To efficiently check for link change, a host always maintains the list of all known prefixes on the link. This procedure of attaining and retaining the Complete Prefix List is initialized when the host is powered on.

The host forms the prefix list at any PoA (Point of Attachment), that is, this process starts independently of any movement. Though the procedure may take some time, that doesn't matter unless the host moves very fast. A host can generate the Complete Prefix List with reasonable certainty if it remains attached to a link sufficiently long. It will take approximately 4 seconds, when it actively performs 1 RS/ RA exchanges. If it passively relies on unsolicited RA messages instead, it may take much more time.

First the host sends an RS to All-Router multicast address. Assuming there is no packet loss, every router on the link would receive the RS and usually reply with an RA containing all the prefixes that the router advertises. However, RFC 2461 mandates certain delays for the RA transmissions.

After an RS transmission, the host waits for all RAs that would have been triggered by the RS. There is an upper limit on the delay of the RAs. MIN_DELAY_BETWEEN_RAS (3 Sec) + MAX_RA_DELAY_TIME (0.5 Sec) + network propagation delay is the maximum delay between an RS and the resulting RAs [1]. 4 seconds would be a safe number for the host to wait for the solicited RAs. Assuming no packet loss, within 4 seconds, the host would receive all the RAs and know all the prefixes. Thus we pick 4 seconds as the value for MAX_RA_WAIT.

In case of packet loss, things get more complicated. In the above process, there may be a packet loss that results in the generation of

the Incomplete Prefix List, i.e. the prefix list that misses some prefix on the link. To remedy this deficiency, the host may perform multiple RS/ RA exchanges to collect all the assigned prefixes.

After one RS/ RA exchange, to corroborate the completeness of the prefix list, the host may send additional RSs and wait for the resulting RAs. The number of RSs is limited to MAX_RTR_SOLICITATIONS [1]. The host takes the union of the prefixes from all the RAs to generate the prefix list. The more RS/ RA exchange the host performs, the more probable that the resulting prefix list is complete. Section 11 gives the detailed analysis.

To ascertain whether its existing prefix list is complete or not, the host can set its own policy. The host may take into consideration the estimated packet loss rate of the link and the number of RS/ RA exchanges it performed or should have performed while it was attached to the link.

In general, the higher the error rate, the longer time and more RA transmissions from the routers are needed to assure the completeness of the prefix list.

3.2 Erroneous Prefix Lists

The host may generate either 1) the Incomplete Prefix List, i.e. the prefix list that does not include all the prefixes that are assigned to the link or 2) the Superfluous Prefix List, i.e. the prefix list that contains some prefix that is not assigned to the link.

It is noted that 1) and 2) are not exclusive. The host may generate the prefix list that excludes some prefix on the link but includes the prefix not assigned to the link.

Severe packet losses during prefix list generation may cause the Incomplete Prefix List. Or the host may have undergone a link change before finishing the procedure of the Complete Prefix List generation. Later we will deal with the case that the host can't be sure of the completeness of the prefix list.

Even if the host falsely assumes that the Incomplete Prefix List is complete, the effect of that assumption is that the host might later think it has moved to a different link when in fact it has not.

In case that a link change happens, even if the host has the Incomplete Prefix List, it will detect a link change. Hence the Incomplete Prefix List doesn't cause a connection disruption. But it may cause extra signaling messages, for example Binding Update messages in [5].

The Superfluous Prefix List presents a more serious problem.

Without the assumed 'link UP' event notification from the link-layer, the host can't perceive that it has changed its attachment point, i.e. it has torn down an old link-layer connection and established a new one. We further discuss the issues, should this assumption be removed, in Section 5.

With the assumed 'link UP' notification, and the assumption of different concurrent layer 2 connections being represented as different (virtual) interfaces to the DNA module (see Section 2.2) the host will never treat RAs from different links as being part of the same link. Hence it will not create a Superfluous Prefix List.

3.3 Link identity detection

When a host receives a hint that indicates a possible link change, it initiates DNA procedure to determine whether it still remains at the same link or not. At this time, the Complete Prefix List generation may or may not be finished.

First, if the host has finished prefix list generation and can be reasonably sure of its completeness, the receipt of a single RA (with at least one valid prefix) is enough to detect the identify of the currently attached link.

Assume that, after the hint, the host receives an RA that contains at least one valid prefix. The host compares the valid prefixes in the RA with those in the existing prefix list. If the RA contains a prefix that is also a member of the existing prefix list, the host is still at the same link. Otherwise, if none of the prefixes in that RA matches the previously known prefixes, it is at a different link.

If the host is not sure that the prefix list was complete before the hint reception, then the host needs to take several RAs into account after the hint reception, before it can determine that it has moved to a different link.

Suppose that before finishing the prefix list generation, the host receives the hint that indicates a possible link change. Then the host can't assume the completeness of the prefix list.

The host can then generate another (complete) prefix list for the (potentially new) link, which compensates for the uncertainty of the old prefix list. After the hint, it performs one or more RS/ RA exchanges additionally to collect all the prefixes on the currently attached link. With the resulting prefixes, the host generates the second prefix list.

Then the host compares two prefix lists and if the lists are disjoint, i.e. have no prefix in common, it assumes that a link change has occurred. Note that if during this procedure, the host finds a common valid prefix between even one RA and the old prefix list, it can immediately determine that it has not moved to a different link.

For example, assume that the host keeps track of how many RS/ RA exchanges it has performed while attached to a link. If this is more than one, i.e. after the host sends one RS and waits 4 seconds for the resulting RAs, the host assumes that it has seen all the prefixes. Suppose that the host doesn't complete even 1 RS/ RA exchange, and then it receives a link UP notification that causes it to initiate the DNA procedure. If the first RA does not have a valid prefix which is common with the old prefixes, then the host needs to wait for additional RAs to complete 1 RS/ RA exchange. In case that the lists are disjoint, the host can assume it has moved.

In summary, first a host makes the Complete Prefix List. When a hint occurs, if the host decides that the prefix list is complete, it will check for link change with just one RA (with a prefix). Otherwise, in case that the host can't be so sure, it will wait for additional RAs to corroborate the decision.

3.4 Renumbering

When the host is sure that the prefix list is complete, a false movement assumption may happen due to renumbering when a new prefix is introduced in RAs at about the same time as the host handles the 'link UP' event. We may solve the renumbering problem with minor modification like below.

When a router starts advertising a new prefix, for the time being, every time the router advertises a new prefix in an RA, it includes at least one old prefix in the same RA. The old prefix assures that the host doesn't falsely assume a link change because of a new prefix. After a while, hosts will recognize the new prefix as the one assigned to the current link and update its prefix list.

In this way, we may provide a fast and robust solution. If a host can make the Complete Prefix List with certainty, it can check for link change fast. Otherwise, it can fall back on a slow but robust scheme. It is up to the host to decide which scheme to use.

4. Protocol Specification

This section provides the actual specification for a host implementing this draft. For generality the specification assumes that the host retains multiple (an unbounded set) of prefix lists until the information times out, while an actual implementation would limit the number of sets maintained.

This description assumes that the link layer driver provides a 'link UP' notification when the host might have moved to a different link.

4.1 Conceptual data structures

This section describes a conceptual model of one possible data structure organization that hosts will maintain for the purposes of DNA. The described organization is provided to facilitate the explanation of how this protocol should behave. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The basic conceptual data type for the protocol is the Candidate Link object. This is an object which contains all the information learned from RA messages that are known to belong to a single link. These data structures are maintained separately for each interface. In particular, this includes

- The valid prefixes learned from the prefix information options, the A/L bits and their valid and preferred lifetimes.
- The default routers and their lifetimes.
- Any other option content such as the MTU etc.

The lifetimes for the prefixes and default routers in the Candidate Link objects should decrement in real time that is, at any point in time they will be the remaining lifetime. An implementation could handle that by recording the 'expire' time for the information, or by periodically decrementing the remaining lifetime.

For each interface, the host maintains a notion of its Current Candidate Link (CCL) object. As we will see below, this might actually be different than the prefix list and default router lists maintained by Neighbor Discovery when the host is in the process of determining whether it has attached to a different link or not.

In addition, the host maintains previous Candidate Link objects. It is per interface since there are some security issues when merging

across interfaces.

The previous Candidate Link objects can be found by knowing at least one prefix that is part of the object.

The operations on Candidate Link objects is to create a new one, discard one, and merge two of them together. The issues with merging are discussed in the next section.

For each interface, the host maintains the last time a valid RA was received (called time_last_RA_received in this document), which actually ignores RAs without prefix options, and the last time a link UP notification was received from the link layer on the host (called time_last_linkUP_received in this document). Together these two conceptual variables serve to identify when a RA containing disjoint prefixes can't be due to being attached to a new link, because there was no link UP notification.

For each interface, the host also maintains a counter (called num_RS_RA) which counts how many successful RS/RA exchanges have been accomplished since the last time the host moved to a different link. The host declares "one successful RS/RA exchange" is accomplished after it sends an RS, waits for MAX_RA_WAIT seconds and receives a positive number of resulting RAs. At least one RA (with at least one prefix) should be received. After the RS, if a link UP event occurs before MAX_RA_WAIT seconds expire, the host should not assume that a successful RS/RA exchange is accomplished. This counter is used to determine when prefix list is considered to be complete. This document considers it to be complete when NUM_RS_RA_COMPLETE (set to 1) number of RS/RA exchanges have been completed.

After one RS/ RA exchange, the host will generate the Complete Prefix List if there is no packet loss. Even some packet loss may cause an Incomplete Prefix List, there is a further chance for the host to get the missing prefixes before it receives link UP notification, i.e. moves to another PoA. Even the host moves to another PoA with Incomplete Prefix List, the first RA may contain the prefix in its prefix list. Considering all those above, even if the host performs only one RS/ RA exchange, it will be rather rare for the host to falsely assume a link change. Moreover, even in case of false detection, there would be no connectivity disruption, because Incomplete Prefix List causes only additional signaling. This document proposes a host to send 1 RS and waits for 4 secs to collect (solicited) RAs and declare CCL complete.

4.2 Merging Candidate Link objects

When a host has been collecting information about a potentially

different link in its Current Candidate Link object, and it discovers that it is in fact the same link as another Candidate Link object, then it needs to merge the information in the two objects to produce a single new object. Since the CCL contains the most recent information, any information contained in it will override the information in the old Candidate Link, for example the remaining lifetimes for the prefixes. When the two objects contain different pieces of information, for instance different prefixes or default routers, the union of these are used in the resulting merged object.

4.3 Timer handling and Garbage Collection

As stated above, the lifetimes for the prefixes and default routers in each Candidate Link object must be decremented in real time. When a prefix' valid lifetime has expired, the prefix should be removed from its object. Likewise, when a default router lifetime has expired, it should be removed from its object. When a Candidate Link object contains neither any prefixes nor any default routers, the object, including additional information such as MTU, should be discarded.

There is nothing to prevent a host from garbage collecting Candidate Link objects before their expire. However, for performance reason a host must be able to retain at least two of them at any given time.

It is recommended to put 90 minute upper limit on how long the objects, other than the CCL, should be retained, to make the protocol more robust against flash renumbering and reassignment.

4.4 Receiving link UP notifications

When the host receives a link UP notification from its link layer, it sets time_last_linkUP_received to the current time.

The host also uses this to trigger sending an RS, subject to the rate limitations in [1]. Since there is no natural limit on how frequently the link UP notifications might be generated, we take the conservative approach that even if the host establishes new link layer connectivity very often, under no circumstances should it send Router Solicitations more frequently than RTR_SOLICITATION_INTERVAL. Thus if it handled the most recent link UP notification less than MAX_RA_WAIT seconds ago, it can not immediately send one when it processes a link UP notification.

If the RS does not result in the host receiving at least one RA with at least one valid prefix, then the host can retransmit the RS. It is allowed to multicast up to MAX_RTR_SOLICITATIONS [1] RS messages spaced RTR_SOLICITATION_INTERVAL apart.

Note that if link-layer notifications are reliable, a host can reset the number of sent Router Solicitations to 0, while still maintaining RTR_SOLICITATION_INTERVAL between RSs. Resetting the count is necessary so that after each link up notification, the host is allowed to send MAX_RTR_SOLICITATIONS to reliably discover the, possibly new, prefix list.

4.5 Receiving valid Router Advertisements

When a host receives a valid RA message (after the validity checks specified in [1]) it performs the following processing in addition to the processing specified in [1] and [2]

If the valid RA does not contain any prefix information options, or all the prefixes have a zero valid lifetime, then no further processing is performed. Note that not even the time_last_RA_received is updated.

If time_last_RA_received is more recent than time_last_linkUP_received, then the host could not possibly have moved to a different link. Hence the only action needed for DNA is to update the current Candidate Link object with the information in the RA, and set time_last_RA_received to the current time. No further processing is performed.

Otherwise, that is if a linkUP indication has been received more recently than time_last_RA_received, we have the case when the host needs to perform comparisons of the prefix sets in its Candidate Link objects and the prefix set in the RA. In this case, time_last_RA_received is always set to the current time.

Should the received RA contain at least one valid prefix which is in the prefix list in the CCL, then the host is still attached to the same link, and just needs to update the CCL with any new information in the RA.

Otherwise, if the received RA contains one or more prefixes which are part of a prefix list in some retained Candidate Link object, then the host has most likely moved back to that link. In this case the host may retain the content of the CCL for future matching, but switch the CCL to be that matching object. The, now new, CCL should be updated based on the information in the RA. Then the DNA module informs the Neighbor Discovery module to replace the old information with the information in the new CCL as specified in Section 4.6.

It is possible that the above comparison will result in matching multiple Candidate Link objects. For example, if the RA contains the prefixes P1 and P2, and there is one Candidate Link object with P1

and P3 and other Candidate Link object with P2 and P4. This should not happen during normal operation, but if links have been renumbered or physically separate links have been made into one link (before the lifetimes in the Candidate Link objects expired), then the host could observe this. One possible action in this case would be for the host to merge all such matching Candidate Link objects together with the information in the received RA and make this the new CCL. Doing this merging correctly requires that each Candidate Link object contains the time it was last updated by a RA, so that more recent information can override older information. The security issues involved in such merging is the prime motivation for not allowing the Candidate Link objects to be shared between different interfaces.

The easy cases of staying on the same link or moving to a previously visited link have been handled above. The harder case is when the first RA after a link UP notification contains prefixes that are new to the host. If the host considers its Current Candidate Link object complete (num_RS_RA is at least NUM_RS_RA_COMPLETE), then an RA where the prefixes are disjoint from those in the CCL, can be assumed to be a link change in accordance with Section 4.6. If the CCL is not considered to be complete, then it isn't obvious whether the host has moved or not, because the CCL might have missed the prefixes in the received RA instead of being associated with a different link. In order to distinguish those two cases the host needs to do some extra work. Thus the host needs to create a new Candidate Link object based on the received RA, and make this object the CCL. However, it does not yet treat this as a new link; it is merely a candidate. Thus it MUST NOT perform the actions in Section 4.6 at this point in time. Instead, the host should wait for MAX_RA_WAIT seconds, and all RAs that are received during that time interval are processed as specified above.

This processing might result in finding a prefix in common between a Candidate Link object and the CCL, in which case the host knows whether and to which link it has moved. But should the MAX_RA_WAIT seconds expire without any common prefix, then it will conclude that it has moved to a new link and inform the rest of the host of the movement (Section 4.6.) Note that the arrival of a new link UP notification during the MAX_RA_WAIT second timer must prevent the MAX_RA_WAIT second timer from firing. In this case the host might yet again have moved so it is necessary to restart the process of inspecting the RAs.

Subject to local policy, and perhaps also the host's knowledge of the packet loss characteristics of the interface or type of L2 technology, the host can try harder than just waiting for MAX_RA_WAIT seconds, by sending additional Router Solicitations. It is allowed to multicast up to MAX_RTR_SOLICITATIONS [1] RS messages spaced

RTR_SOLICITATION_INTERVAL apart. In the most conservative approach this means a 12 second delay until the host will declare that is has moved to a new link. Just as above, this process should be terminated should a new link UP notification arrive during the 12 seconds.

4.6 Changing the link in Neighbor Discovery

When DNA detects that it has moved to a different link this needs to cause Neighbor Discovery, Address autoconfiguration, and DHCPv6 to take some action. While the full implications are outside of the scope of this document, here is what we know about the impact on Neighbor Discovery.

Everything learned from the RAs on the interface should be discarded, such as the default router list and the on-link prefix list. Furthermore, all neighbor cache entries, in particular redirects, need to be discarded. Finally the information in the Current Candidate Link object is used to create a new default router list and on-link prefix list.

The list of things are potentially affected by this movement is fairly extensive, since new Neighbor Discovery options are being created. In addition to what is mentioned above, the list includes:

- The MTU option defined in [1].
- The Advertisement Interval option defined in [5].
- The Home Agent Information option defined in [5].
- The Route Information option defined in [11].

In addition, when the host determines it has moved it needs to set num_RS_RA to zero.

5. CPL without a 'link UP' notification

If the host implementation does not provide any link-layer event notifications [9], and in particular, a link UP notification, the host needs additional logic to try to decide whether a received RA applies to the "old" link or a "new" link.

In this case there is an increased risk that the host get confused, thus it isn't clear whether this should be part of the recommendation, or whether we should just require that hosts which implement this draft have a 'link UP' notification.

As the protocol is specified in Section 4, if there is no 'link UP' notification when the host might have moved, the host would collect the prefixes from multiple links into a single Candidate Link object, and would never detect movement.

Here is an example. The host begins to collect the prefixes on a link. But before the prefix list generation is completed, without its knowledge, the host moves to a new link. Unaware that now it is at the different link, the host keeps collecting prefixes from the received RAs to generate the prefix list. This results in the prefix list containing prefixes from two different links. If the host uses this prefix list, it fails to detect a link change.

A possible way to prevent this situation for implementations without a link UP notification, is to treat the arrival of a RA with a disjoint set of prefixes as a hint, the same way Section 4 treats the link UP notification as a hint, as specified below.

The implications of treating such an RA as a hint, is that such an RA would set 'time_last_linkUP_received' to the current time, create a new Candidate Link object with the information extracted from that RA, and then send an RS as specified in Section 4.4.

However, there is still a risk for confusion because the host can not tell from the RAs whether they were solicited by the host. (RFC 2461 recommends that solicited RAs be multicast.) The danger is exemplified by this:

1. Assume the host has a CCL with prefixes P1 and P2.

2. The host changes link layer attachment, but there is no link UP notification.

3. The host receives an RA with a disjoint set of prefixes: prefix P3. This causes the host to form a new Candidate Link object with P3 and send an RS.

4. The host again changes link layer attachment, and no link UP notification.

5. The host receives one of the periodic multicast RAs on the link, which contains prefix P4. It can not tell whether this RA was in response to the RS it send above. The host ends up adding this to the CCL, which now has P3 and P4, even though those prefixes are assigned to different links.

There doesn't appear to be a way to solve this problem without changes to the routers and the Router Advertisement messages. However, the probability of this occurring can be limited by limiting the window of exposure. The simplest approach is for the host to assume that any RA received within MAX_RA_WAIT seconds after sending an RS was in response to the RS. Basically this relies on the small probability of both moving again in that MAX_RA_WAIT second interval, and receiving one of the periodic RAs. If the periodic RAs are sent infrequently enough, this might work in practise, but is by no means bullet-proof.

6. IANA Considerations

No new message formats or services are defined in this document.

7. Security Considerations

DNA process is intimately related to Neighbor Discovery protocol and its trust model and threats have much in common with the ones presented in RFC 3756 [7]. Nodes connected over wireless interfaces may be particularly susceptible to jamming, monitoring, and packet insertion attacks. Use of [6] to secure Neighbor Discovery are important in achieving reliable detection of network attachment. DNA schemes SHOULD incorporate the solutions developed in IETF SEND WG if available, where assessment indicates such procedures are required.

The threats specific to DNA are that an attacker might fool a node to detect attachment to a different link when it is in fact still attached to the same link, and conversely, the attacker might fool a node to not detect attachment to a new link.

The first form of attack is not very serious, since at worst it would imply some additional higher-level signaling to register a new (care-of) address. The second form of attack can be more serious, especially if the attacker can prevent a host from detecting a new link. The protocol as specified would require an attacker to be on-link and be authenticated and authorized to send Router Advertisements when Secure Neighbor Discovery [6] is in use. However, even without SEND, an attacker would need to send RAs containing the prefixes to which it wants the host to be unable to detect movement. This can be done for a small number of prefixes, but it isn't possible for the attacker to completely disable DNA for all possible prefixes on other links.

8. Examples

This section contains some example packet flows showing the operation of prefix based DNA.

8.1 Example with link UP event notification

Assume the host has seen no link UP notification for a long time and that it has the prefixes P1, P2, and P3 in its prefix list for the interface.

The IP layer receives a link UP notification. This hint makes it multicast an RS and start collecting the received prefixes in a new list of prefixes.

The host receives an RA containing no prefixes. This has no effect on the algorithm contained in this specification.

The host receives an RA containing only the prefix P4. This could be due to being attached to a different link or that there is a new prefix on the existing link which is not announced in RAs together with other prefixes, and a spurious hint. In this example the host decides to wait for another RA before deciding.

One second later an RA arrives which contains P1 and P2. As a result the "new" prefix list has P1, P2, and P4 hence is not disjoint from the "old" prefix list with P1, P2, and P3. Thus the host concludes it has not moved to a different link and its prefix list is now P1, P2, P3, and P4.

Some time later a new link UP notification is received by the IP layer. Triggers sending a RS.

An RA containing P5 and P6 is received by the host. Based on some heuristic (for instance, the number of RAs it received on the old link, or the assumed frequency of prefixes being added to an existing link) this time the host decides that it is on a new link.

One second later an RA with prefix P7 is received. Thus the prefix list now contains P5, P6, and P7.

8.2 Example without link UP event notification

Assume the host has collected the prefixes P1, P2, and P3 in its prefix list for the interface.

The host receives an RA containing only prefix P4. The fact that P4 is disjoint from the prefix list makes this be treated as a hint.

This hint makes the host multicast an RS and start collecting the received prefixes in a new list of prefixes, which is initially set to contain P4.

The host receives an RA containing no prefixes. This has no effect on the algorithm contained in this specification.

The host receives an RA containing only the prefix P4. This could be due to being attached to a different link or that there is a new prefix on the existing link which is not announced in RAs together with other prefixes. In this example the host decides to wait for another RA before deciding.

One second later an RA arrives which contains P1 and P2. As a result the "new" prefix list has P1, P2, and P4 hence is not disjoint from the "old" prefix list with P1, P2, and P3. Thus the host concludes it has not moved to a different link and its prefix list is now P1, P2, P3, and P4.

Some time later the host receives an RA containing prefix P7. This is treated as a hint since it is not part of the current set of prefixes. Triggers sending a RS and initializing the new prefix list to P7.

An RA containing P5 and P6 is received by the host. This is disjoint with both of the previous prefix lists, thus the host might be attached to a 3rd link after very briefly being attached to the link with prefix P7. The host decides to wait for more RAs.

One second later an RA with prefix P7 is received. It still isn't certain whether P5, P6, and P7 are assigned to the same link (and without a link UP notification such uncertainties do exist).

A millisecond later an RA with prefixes P6 and P7 is received. Now the host decides that P5,P6, and P7 are assigned to the same link.

Four seconds after the RS was sent and no RA containing P1, P2, P3, or P4 has been received the host can conclude with high probability that it is no longer attached to the link which had those prefixes.

9. Protocol Constants

The following protocol constants are defined in this document.

Constant name	Constant value
NUM_RS_RA_COMPLETE	1
MAX_RA_WAIT	4 seconds

Table 1

10. Acknowledgements

The authors would like to acknowledge the many careful comments from Greg Daley that helped improve the clarity of the document, as well as the review of the DNA WG participants in general.

11. Performance Analysis

In this section, we compute the probability that a host fails to generate the Complete Prefix List due to packet loss, and consequently assumes a link change when the host in fact did not move to a different link.

Suppose, in a link, there are N routers, R[1], R[2],...., R[N].

Each R[i] advertises the Router Advertisement RA[i] with the prefix P[i].

It is the worst case that each router advertises the different prefix. It is necessary to receive all the RA[i] to generate the Complete Prefix List.

We assume there is a host, H, and when the host sends a Router Solicitation, let P be the probability that it fails to receive a RA[i] because of a RA loss. For the simplicity, we disregard RS losses.

So when the sends a Router Solicitation, the probability that it will receive all RA[i] is (1-P)^N.

Let's assume the host performs RS/ RA exchange T times, 1,2,..,T.

Let S[k] be the set of all RAs which the host H successfully receives at k-th RS/RA exchange. The probability that R[i] belongs to S[k] is (1-P).

Let PL[k] be the set of prefixes which are made from S[k], i.e. the set of P[j] such that RA[j] belongs to S[k]. Obviously, the probability that P[i] belongs to PL[k] is also (1-P).

Let PL be the union of all PL[k], from k=1 to k=T. PL is the prefix list made from performing RS/ RA exchange T times.

1) The probability of the Complete Prefix List generation

First the probability that P[i] belongs to PL is 1-P^T. The probability that the prefix list PL is complete is (1-P^T)^N.

For example, assume the error rate is 1 % and there are 3 routers in a link, then, with 2 RS/ RA exchanges, the probability of generating an accurate Complete Prefix List is roughly 99.97 %.

At this point, assume that the host H receives a hint that a link change might have happened and consequently initiates the procedure

of checking a link change.

2) The false DNA probability if the host checks for link change with one RA.

Assume one RA, whether solicited or unsolicited arrives. If the host H makes a decision based solely on the RA and the prefix list, the probability that it falsely assume a link change is P^T.

For example, given the error rate is 1%, with 2 RS/ RA exchanges, the probability of false movement detection is 1/ 10000.

3) The false DNA probability if the host checks for link change with additional RS/ RA exchanges.

Instead of depending on the single RA, the host H performs additional RS/ RA exchange U times, 1,2...U. Then the probability that H falsely assumes a link change is

[P^T + P^U - P^(T+U)]^N.

For example, given the error rate is 1 % and there are 3 routers in a link, if the host H performs 2 RS/ RA exchanges before the hint and 1 RS/ RA exchange after one, the probability of false movement detection is roughly 1/1000000.

In the above formula, the result goes to P^(U*N) as T goes infinity. The term P^(U*N) results from the probability that the host receives no RA during U RS/ RA exchange after the hint. To see that it still remains at the same link, a host needs to receive at least one RA.

We think it is reasonable to assume that the RS will be retransmitted until at least one RA arrives. If we take a one more assumption that the host receives at least one RA, the probability will be

[[P^T + P^U - P(T+U)]^N - P^(U*N)]/ [1- P^(U*N)]

The above converges to zero as T approaches infinity.

12. Change Log

The following changes have been made since draft-ietf-dna-cpl-01:

- A few editorial changes. For example, we use the term 'PoA (Point of Attachment)' instead of attachment point in accordance with DNAv4 draft [12].
- Clarify further that a host is recommended to declare its prefix list complete after 1 RS/ RA exchange. We also added a text about why it is recommended such in section 4.1.
- Make the definition of "successful exchange" more precise in section 4.1.

The following changes have been made since draft-ietf-dna-cpl-00:

- Many editorial fixes
- Added a count to the CCL to track whether it is likely to be complete (num_RS_RA)
- Set the default threshold for this count to 1, that is, after a single RS/RA exchange that resulted in at least one RA being received with a useful prefix, the prefix list will be considered to be complete. The value is named NUM_RS_RA_COMPLETE.
- In section 4.5 added some fudge around whether merging when a RA has prefixes which matches multiple Candidate Link objects. We need to decide what to specify in this area.
- Clarified section 4.5 that Candidate Link objects can not be shared between different interfaces.

The following changes have been made since draft-jinchoi-dna-cpl-01:

- Clarified that only prefixes with a non-zero valid lifetime are considered.
- Added some text about renumbering considerations.
- Limited the retention of old Candidate Link objects to 90 minutes to avoid problems if there is flash renumbering *and* a prefix is reassigned to a different link in less than 90 minutes.
- Explicitly made the assumption that the host implementation has a 'link UP' event notification.

- Added missing text in section 4.4 about sending a RS when a link UP notification is processed.

- Added text in section 4.6 to say that current and future ND options need to be included in the information that is discarded when the host declares that is has moved to a different link.

- Made the Candidate Link objects be per interface, since there are some security issues when they are shared between interfaces that might be of different trustworthyness.

- Many editorial clarifications.

13. Open Issues

- Should we worry about implementations without 'link Up' notifications? The technique in Section 5 is far from bullet-proof.

- Flash renumbering and immediate reassignment may cause a problem. Assume a prefix is suddenly removed from one link and immediately reassigned to an another link. A host in first link may not perceive the prefix removal and mistakenly assume the prefix is still valid. If the host moves to the second link and check for link change with the prefix, it will make a false decision.

14. References

14.1 Normative References

[1] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

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

[3] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[4] Choi, JH. and G. Daley, "Goals of Detecting Network Attachment in IPv6", RFC 4135, August 2005.

14.2 Informative References

[5] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[6] Arkko, J., Kempf, J., Sommerfeld, B., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", draft-ietf-send-ndopt-06 (work in progress), July 2004.

[7] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[8] Choi, J. and E. Nordmark, "DNA solution framework", draft-jinchoi-dna-soln-frame-00 (work in progress), July 2004.

[9] Yegin, A., "Link-layer Event Notifications for Detecting Network Attachments", draft-ietf-dna-link-information-03 (work in progress), October 2005.

[10] Pentland, B., "An Overview of Approaches to Detecting Network Attachment in IPv6", draft-dnadt-dna-discussion-00 (work in progress), February 2005.

[11] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", draft-ietf-ipv6-router-selection-07 (work in progress), January 2005.

[12] Aboba, B., "Detecting Network Attachment in IPv4 (DNAv4)", draft-ietf-dhc-dna-ipv4-18 (work in progress), December 2005.

Authors' Addresses

JinHyeock Choi
Samsung AIT
Communication Lab
P.O.Box 111 Suwon 440-600
KOREA

Phone: +82 31 280 9233
Email: jinchoe@samsung.com

Erik Nordmark
Sun Microsystems
17 Network Circle
Menlo Park, CA 94043
USA

Phone: +1 650 786 2921
Email: erik.nordmark@sun.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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