Network Working Group
Internet-Draft
Expires: April 18, 2005

J. Arkko
Ericsson

October 18, 2004

Failure Detection and Locator Selection in Multi6
draft-arkko-multi6dt-failure-detection-00

Status of this Memo

Copyright Notice

Abstract

This draft discusses locator pair selection and failure detection mechanisms for the IPv6 multihoming feature being developed in the Multi6 working group. Elements of this document may also be useful for developing the details of the MOBIKE or HIP multihoming mechanisms. The draft also discusses the roles of a multihoming protocol versus network attachment functions at IP and link layers.

Table of Contents

1. Introduction

The Multi6 working group is extending IPv6 to support multihoming. A number of possible approaches exist in this space, but the current focus of the group is to look at an IP layer (or layer 3.5) mechanism that hides multihoming from applications. Different variants of the IP layer mechanism have been suggested in [17, 18, 19, 21] and other references.

All these mechanisms have a common need to detect when a switch to another address or addresses becomes necessary. We call this failure detection, because the multi6 protocol works primarily as a failover rather than a load balancing scheme.

This draft discusses what requirements such a component of the multi6

protocol has, and how these requirements can be achieved. The draft is structured as follows: Section 2 discusses what kind of solutions have been used in other similar protocols. Section 3 defines a set of useful terms and discusses them, and xref target='transport'/> discusses the architectural implications of multihoming at IP layer. Finally, Section 5 describes one possible solution involving two state machines, a failure testing protocol, and an address pair selection algorithm.

For the purposes of this draft, we consider an address to be synonymous with a locator. There may be other, higher level identifiers such as security associations, FQDNs, CGA public keys, or HITs that tie the different locators used by a node together.

2. Related Work

In SCTP [9], the addresses of the endpoints are learned in the connection setup phase either through listing them explictly or via

giving a DNS name that points to them. In order to provide a failover mechanism between multihomed hosts, SCTP has the following functions:

- One of the peer's addresses is selected as the primary address by the application running on top of SCTP. All data packets are sent to this address until there is a reason to choose another address, such as the failure of the primary address.

- Testing the reachability of the peer endpoint's addresses. This is done both via observing the data packets sent to the peer or via a periodic heartbeat when there is no data packets to send.

  Each time data packet retransmission is initiated (or when a heartbeat is not answered within the estimated round-trip time) an error counter is incremented. When a configured error limit is reached, the particular destination address is marked as inactive. The reception of an acknowledgement or heartbeat response clears the counter.

- Retransmission: When retransmitting the endpoint attempts pick the most "divergent" source-destination pair from the original source-destination pair to which the packet was transmitted. Rules for such selection are, however, left as implementation decisions in SCTP.

SCTP does not define how local knowledge (such as information learned from the link layer) should be used. SCTP also has no mechanism to deal with dynamic changes to the set of available addresses.

The MOBIKE protocol is currently being designed, and some proposals for the protocol exists [12, 13, 14, 15]. No official decision about the protocol has been made yet, but there has been a lot of discussion around the failure detection mechanisms in the context of MOBIKE, and reference [10] records some of the current thoughts of the WG on this issue.

Some of the issues that have been discussed include the following:

- Single address vs. multiple peer addresses. A simple approach is to have the peers be aware of just the current address of the other side instead of all possible ones. Assuming that one of the peers will request the other to start sending to a new address

this works well. However, this approach is unable to deal with

problems that affect both nodes. For instance, two nodes connected by two separate point-to-point links will be unable to switch to the other link if a failure occurs on the first one.

- Addresses vs. address pairs. Are tests and current paths individual peer addresses, or pairs of peer and own addresses (paths)? It seems that some failure scenarios require the use of a path rather than a single address. A network failure may make it impossible to communicate between a particular pair of addresses, even if those addresses have some other connectivity.

- Where the connectivity information comes from. Does it come from local stack (such as interface up/down, router advertisement), from reception of ESP packets, from IKEv2 keepalives, or through some MOBIKE-defined mechanism?

The mobility and multihoming specification for the HIP protocol [16] leaves the determination of when address updates are sent to a local policy, but suggests the use of local information and ICMP error messages.

Network attachment procedures are also relevant for multihoming. The IPv6 and MIP6 working groups have standardized mechanisms to dynamically learn about new networks that a node has attached to, and enhanced or optimized mechanisms are being designed in the DHC and DNA working groups. Network attachment detection has turned out to be a relatively complex procedure for mobile hosts, and it was not fully anticipated at the time IPv6 Neighbor Discovery or DHCP were being designed.

3. Definitions

This section defines terms useful in discussing the failure detection problem space.

3.1 Available Addresses

Multi6 nodes need to be aware of what addresses they themselves have. If a node loses the address it is currently using for communications, another address must replace this address. And if a node loses an address that the node's peer knows about, the peer must be informed. Similarly, when a node acquires a new address it may generally wish the peer to know about it.

Definition. Available address. An address is said to be available if the following conditions are fulfilled:

- The address has been assigned to an interface of the node.

- If the address is an IPv6 address, we additionally require that (a) the address is valid in the sense of RFC 2461 [2], and that (b) the address is not tentative in the sense of RFC 2462 [3]. In other words, the address assignment is complete so that communications can be started.

  Note this explicitly allows an address to be optimistic in the sense of [7] even though implementations are probably better off using other addresses as long as there is an alternative.

- The address is a global unicast or unique site-local address [8]. That is, it is not an IPv6 link-local or site-local address. Where IPv4 is considered, it is not an RFC 1918 address.

- The address and interface is acceptable for use according to a local policy.

Available addresses are discovered and monitored through mechanisms outside the scope of MULTI6 (and HIP or MOBIKE). These mechanisms include IPv6 Neighbor Discovery and Address Autoconfiguration [2, 3], DHCP [4], enhanced network detection mechanisms detected by the DNA working group, and corresponding IPv4 mechanisms, such as [6].

3.2 Locally Operational Addresses

Two different granularity levels are needed for failure detection. The coarser granularity is for individual addresses:

Definition. Locally Operational Address. An available address is

said to be locally operational when its use is known to be possible locally: the interface is up and the relevant default router (if applicable) is known to be reachable.

Locally operational addresses are discovered and monitored through mechanisms outside MULTI6 (and HIP or MOBIKE). These mechanisms include IPv6 Neighbor Discovery [2], corresponding IPv4 mechanisms, and link layer specific mechanisms. Theoretically, it is also possible for hosts to learn about routing failures for a particular selected source prefix, even if no protocol exists today to distribute this information in a convenient manner.

3.3 Operational Address Pairs

The existence of locally operational addresses are not, however, a guarantee that communications can be established with the peer. A failure in the routing infrastructure can prevent the sent packets from reaching their destination. For this reason we need the definition of a second level of granularity, for pairs of addresses:

Definition. Bidirectionally operational address pair. A pair of locally operational addresses are said to be an operational address pair, iff bidirectional connectivity can be shown between the addresses. That is, a packet sent with one of the addresses in the source field and the other in the destination field reaches the destination, and vice versa.

Unfortunately, there are scenarios where bidirectionally operational address pairs do not exist. For instance, ingress filtering or network failures may result in one address pair being operational in one direction while another one is operational from the other direction. The following definition captures this general situation:

Definition. Undirectionally operational address pair. A pair of locally operational addresses are said to be an unidirectionally operational address pair, iff packets sent with the first address as the source and the second address as the destination can be shown to reach the destination.

Both types of operational pairs are discovered and monitored through the following mechanisms:

- Positive feedback from upper layer protocols. For instance, TCP can indicate to the IP layer that it is making progress. This is similar to how IPv6 Neighbor Unreachability Detection can in some cases be avoided when upper layers provide information about

bidirectional connectivity [2]. In the case of unidirectional connectivity, the upper layer protocol responses come back using

another address pair, but show that the messages sent using the first address pair have been received.

- Negative feedback from upper layer protocols. It is conceivable that upper layer protocols give an indication of a problem to the MULTI6 layer. For instance, TCP could indicate that there's either congestion or lack of connectivity in the path because it is not getting ACKs.

- Explicit reachability tests. For instance, the IKEv2 keepalive mechanism can be used to test that the current pair of addresses is operational.

- ICMP error messages. Given the ease of spoofing ICMP messages, one should be careful to not trust these blindly, however. Our suggestion is to use ICMP error messages only as a hint to perform an explicit reachability test, but not as a reason to disrupt ongoing communications without other indications of problems.

Note that some protocols, such as HIP [16], perform a return routability test of an address before it is taken into use. The purpose of this test is to ensure that fraudulent peers do not trick others into redirecting traffic streams onto innocent victims [22]. Such tests can at the same time work as a means to ensure that an address pair is operational. Note, however, that some advanced optimizations attempt to postpone the reachability tests so that they do not increase movement-related latency [20].

3.4 Primary Address Pair

Contrary to SCTP which has a specific congestion avoidance design

suitable for multi-homing, IP-layer solutions need to avoid sending packets concurrently over multiple paths; TCP behaves rather poorly in such circumstances. For this reason it is necessary to choose a particular pair of addresses as the primary address pair which is used until problems occur, at least for the same session.

A primary address pair need not be operational at all times. If there is no traffic to send, we may not know if the primary address pair is operational. Neverthless, it makes sense to assume that the address pair that worked in some time ago continues to work for new communications as well.

3.5 Miscellaneous

Addresses can become deprecated [2]. When other operational addresses exist, nodes generally wish to move their communications away from the deprecated addresses.

Similarly, IPv6 source address selection [5] may guide the selection of a particular source address - destination address pair.

4. Architectural Considerations

Architecturally, a number of questions arises. One simple question is whether there needs to be communications between a multihoming solution residing at the IP layer and upper layer protocols? Upon changing to a new address pair, transport layer protocol SHOULD be notified so that it can perform a slow start. This is necessary, for instance, when switching from a high-bandwidth LAN interface to a low bandwidth cellular interface. (Note that this notification can not

be done in protocol designs where the end points are not the final hosts, such as where a gateway is used.

A more fundamental question is which protocols should be responsible for which parts of the problem. It seems clear that no multihoming solution should take on the task of lower layers and other IP functions for discovering its own addresses or testing local connectivity. Protocols such as DHCP or Neighbor and Router Discovery do this already.

But it is less clear which protocol(s) should discover end-to-end connectivity problems or recover from them. One answer is that this is clearly within the domain of multihoming protocol. By performing testing and failure detection of the used path and switching to a new path if necessary, the transport and application protocols can work unchanged.

On the other hand, one could argue that transport and application protocols would have more knowledge about the situation, and have a better ability to decide when a move is required. For instance, they know what the required throughput and congestion status is. Also, it would be unfortunate if both the IP layer and transport/application layer took action for the same problem, for instance by switching to a new address at the IP layer and throttling back due to "congestion" at the transport layer.

Generally speaking, we can divide information that a host has into three categories: local information from "lower layers" such as IPv6 Neighbor Discovery, transit and congestion condition information from either from the multihoming protocol itself or from transport layer protocols and (where available) ECN, and application layer policies that dictate what the requirements are for acceptable connections.

The division of work is largely left as an open issue as far as this document is concerned, but our description works from a point of view of a multihoming protocol at the IP layer. We also note that in the CELP proposal [11], both IP, transport, and application layer entities could share their connectivity status in a common information pool. This may also be a useful approach.

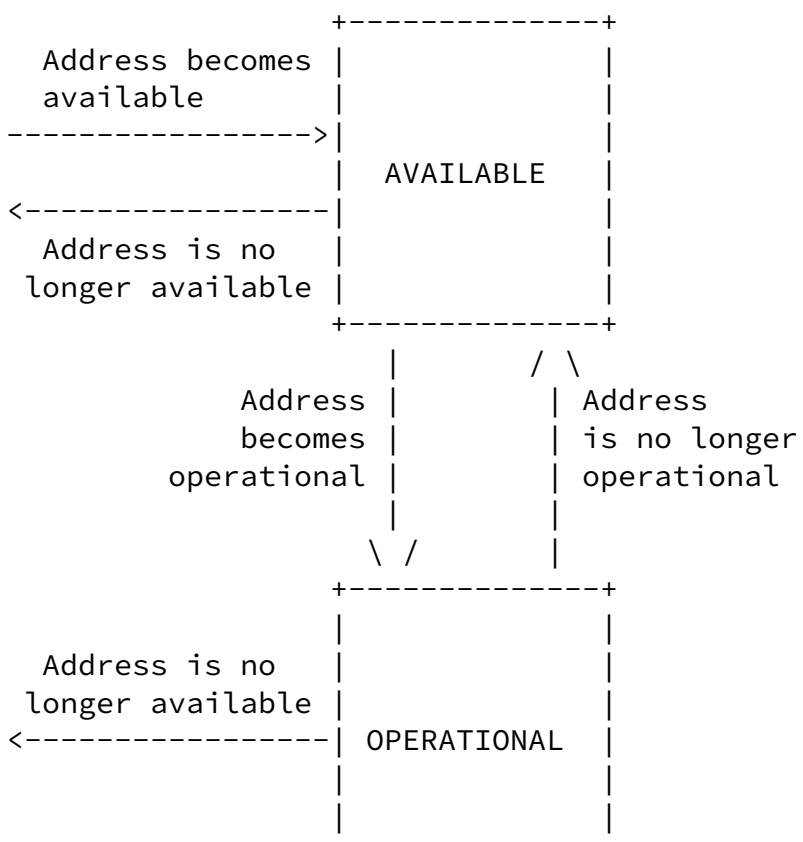
Finally, the last architectural question is about the difference between mobility and multihoming. Given our definitions above, there's no fundamental difference with respect to how the multihoming/mobility protocol learns the addresses it has available. However, a practical difference is that in a multihoming scenario there are alternative addresses, whereas in mobility changes to a new address are forced due to the old address no longer being available.

5. An Approach

One suggested approach consists of a mechanism for keeping track of the host's own available addresses, operational addresses, and operational address pairs.

5.1 State Machine for Addresses

Addresses can be in the AVAILABLE and OPERATIONAL states. The state transitions relating to this are shown in Figure 1.



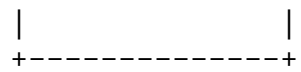


Figure 1. Address state machine.

When an address becomes operational, it SHOULD be reported as a new address to the peer. Similarly, when an address is no longer operational or available, the peer SHOULD be informed.

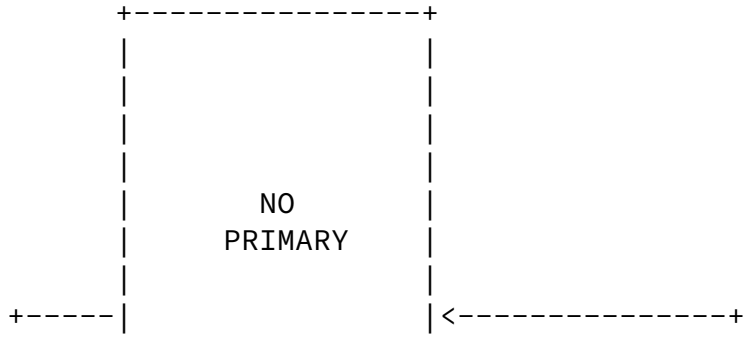
In addition, a particular address can be either preferred or deprecated. This is not shown in the state machine.

5.2 State Machine for Address Pair Selection

A node runs the address pair selection state machine to choose the currently used primary address pair, the one which is used for sending outgoing packets. A node runs one of these state machines towards each different peer, tracking the known address pairs and their status. Each peer also has its own state machine for talking back to the node; there is no guarantee that the same address pairs (in reverse order) have the same state; lack of bidirectionally operational pair would result in a different state on both sides, for instance.

The state machine can be in the NO PRIMARY, TESTING PRIMARY, and PRIMARY OPERATIONAL states. The chosen address pair is known to be operational in the PRIMARY OPERATIONAL state, and is either unverified or non-operational in the other states.

Figure 2 shows the state machine:


NO
PRIMARY

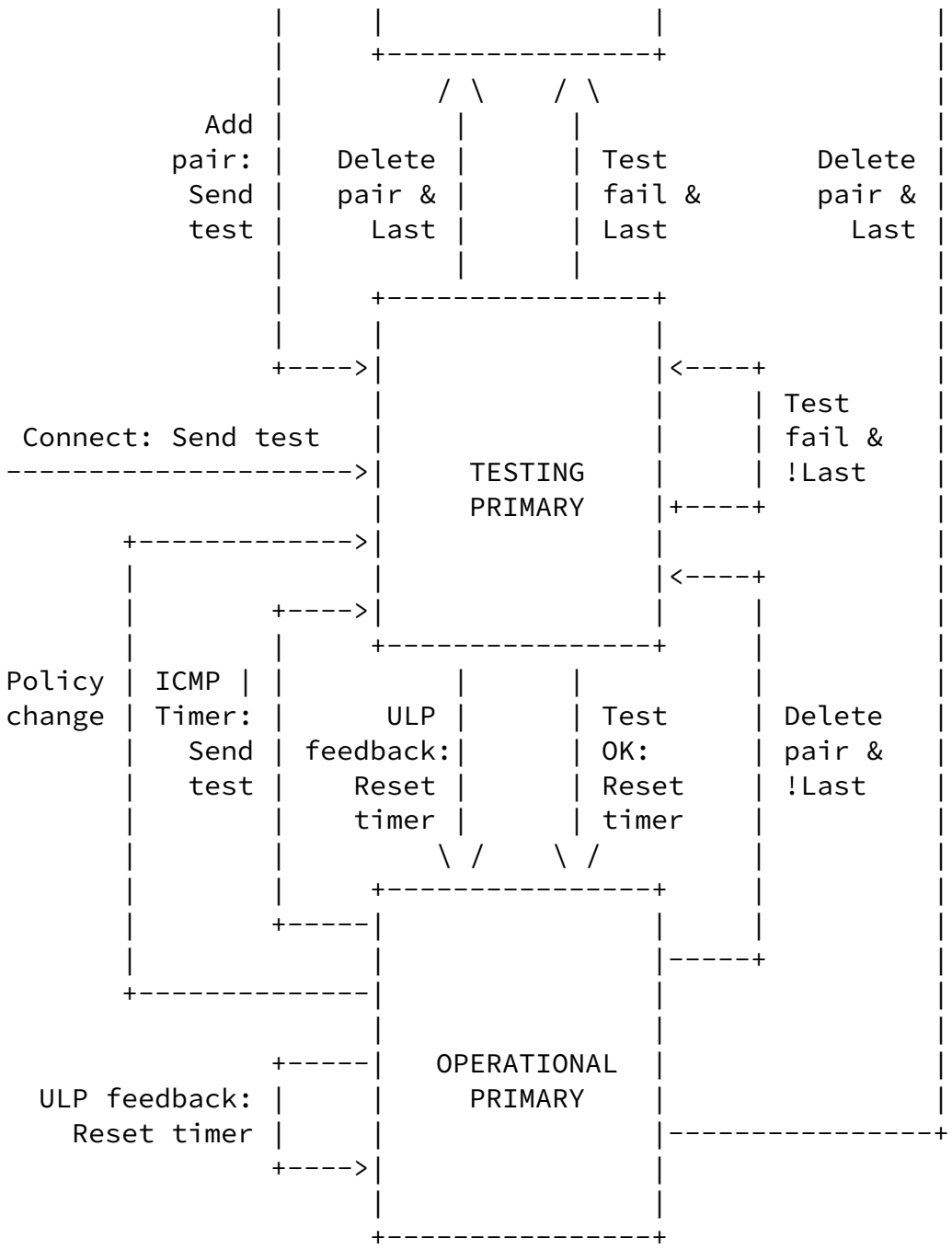


Figure 2. Pair selection state machine.

The notation used in Figure 2 is explained below:

Connect

: An event representing the desire of the application to send a packet to a new peer, or an indication from a peer wishing to connect to us.

Test OK

: An event representing a successful completion of the reachability test.

Test fail

: An event representing failure to complete the reachability test.

ULP feedback

: An event representing positive indication from an upper layer protocol that the packets we have sent to the peer are getting through.

ICMP

: An event representing the reception of an ICMP error message.

Timer

: An event representing timer elapsing.

Add pair

: An event representing the addition of a new possible address pair, either through learning a new local address or being told of a new remote address.

Delete pair

An event representing the deletion of the currently chosen primary address pair.

Policy change

An event representing the desire of the local or remote end to change to a different address pair, despite the current one being operational. This can be due to the availability of the higher-bandwidth connection, cost, or other issues.

Last

A condition that tells whether or not the currently chosen primary pair is the only known address pair.

Send test

An action to initiate the reachability test for a particular pair. This test is typically embedded in the Multi6 connection setup exchange when run initially, and a separate exchange later.

Note that due to potentially asymmetric connectivity, both sides have to perform their own tests, and make their own primary pair selections.

An action to reset a timer so that it will send an event after a specified time.

The state machines also assumes an underlying multihoming signaling

capabability, consisting of the following abstract message exchanges:

Open

Establishes a connection between the peers. May also exchange locator sets and test reachability at the same time.

Test

Verifies reachability using a specific address pair.

Add

Informs the peer about new locators.

Delete

Informs the peer about losing some locators.

Note that the above state machine leaves open how specific address pairs are chosen, as this will be discussed in the next section. We have also, on purpose, decided to avoid attaching functional labels such as "backup" to other address pairs beyond the primary pair. It is our belief that a general design does not need these labels.

5.3 Pair Selection Algorithm

The pair selection state machine assumes an ability to pick primary and alternative address pairs.

This process result in a combinatorial explosion when there are many addresses on both sides. Do both sides track all possible combinations of addresses? If a failure occurs, shall all combinations be tested before giving up? Are such tests performed in parallel or in sequence, and what kind of backoff procedures should be applied?

Our suggestion is that nodes MUST first consult RFC 3484 [5] policy tables to determine what combinations of addresses are legal from a local point of view, as this reduces the search space. Nodes SHOULD also use local information, such as known quality of service parameters or interface types to determine what addresses are preferred over others, and try pairs containing such addresses first. In some cases we can also learn the peer's preferences through the multihoming protocol [16].

> Discussion note 1: It may also be possible to simulate preferences by choosing to not tell the peer about some (non-preferred) addresses.

> Discussion note 2: The preferences may either be learned dynamically or be configured. It is believed, however, that dynamic learning based purely on the MULTI6 protocol is too hard and not the task this layer should do. Solutions where multiple protocols share their information in a common pool of locators could provide this information from transport protocols, however [11].

The reception of packets from the peer with a given address pair is a good hint that the address pair works, particularly when these packets are authenticated multihoming protocol packets. However, the reception of these packets alone is an insufficient reason to switch to a new address, as in an unidirectional connectivity case the return path may not work.

One suggested good implementation strategy is to record the reachability test result (an on/off value) and multiply this by the age of the information. This allows recently tested address pairs to be chosen before old ones.

Out of the set of possible candidate address pairs, nodes SHOULD attempt a test through all of them, but MUST do this sequentially (based on an implementation-dependent priority order) and using an exponential back-off procedure.

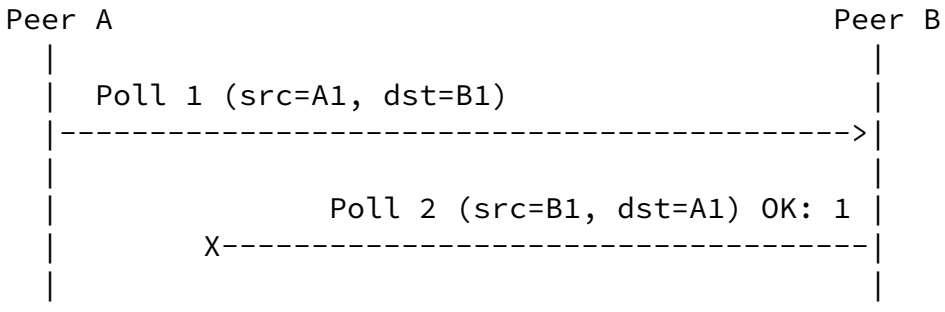
This sequantial process is necessary in order to avoid a "signaling storm" when an outage occurs (particularly for a complete site). However, it also limits the number of addresses that can in practice

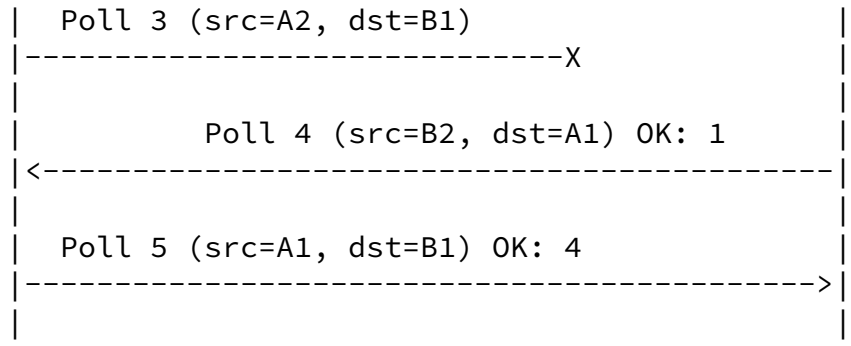
be used for multihoming, considering that transport and application layer protocols will fail if the switch to a new address pair takes too long. For instance, we can assume that an initial timeout value is 0.1 seconds and there are four addresses on both sides. Going through all sixteen address pairs and doubling the timeout value at every trial would take 3200 seconds!

Finally, as has been noted in the context of MOBIKE, the existence of NATs can require that peers continuously monitor the operational status of address pairs, as otherwise NAT state related to a particular communication is lost, and the peer on the outer side of the NAT can no longer reach the peer inside the NAT.

5.4 Protocol for Testing Unidirectional Reachability

Testing for reachability is not easy in an environment where unidirectional reachability is a possibility. This is because the test of a single pair may not result in a working paths to send both the request and response packets. The following protocol could be used to avoid this problem:





When B receives the first Poll message, it memorizes that it has gotten it. The Poll message from B, however, is lost so A tries again with another pair. This is lost too, but B continues its own testing process by sending its second Poll message, which is received by A. The messages carry identifiers, and a list of identifiers that were found messages the sender had itself successfully received earlier.

In the end of the example case, A and B know that they have a working path from A to B using (A1, B1) and from B to A using (B2, A1).

More generally, when A decides that it needs to test for connectivity, it will initiate a set of Poll messages, in sequence, until it gets a Poll message from B indicating that (a) B has received one of A's Poll messages and, obviously, (b) that B's Poll message is getting through. B uses the same algorithm, but starts the process from the reception of the first Poll mesage from A.

6. References

6.1 Normative References

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

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

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

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

[5] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[6] Aboba, B., "Detection of Network Attachment (DNA) in IPv4", draft-ietf-dhc-dna-ipv4-08 (work in progress), July 2004.

[7] Moore, N., "Optimistic Duplicate Address Detection for IPv6", draft-ietf-ipv6-optimistic-dad-01 (work in progress), June 2004.

[8] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-unique-local-addr-05 (work in progress), June 2004.

6.2 Informative References

[9] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[10] Kivinen, T., "Design of the MOBIKE protocol", draft-ietf-mobike-design-00 (work in progress), June 2004.

[11] Crocker, D., "Framework for Common Endpoint Locator Pools", draft-crocker-celp-00 (work in progress), February 2004.

[12] Dupont, F., "Address Management for IKE version 2", draft-dupont-ikev2-addrmgmt-05 (work in progress), June 2004.

[13] Eronen, P., "Mobility Protocol Options for IKEv2 (MOPO-IKE)",

draft-eronen-mobike-mopo-00 (work in progress), July 2004.

[14] Eronen, P. and H. Tschofenig, "Simple Mobility and Multihoming Extensions for IKEv2 (SMOBIKE)", draft-eronen-mobike-simple-00 (work in progress), March 2004.

[15] Kivinen, T., "MOBIKE protocol", draft-kivinen-mobike-protocol-00 (work in progress), March 2004.

[16] Nikander, P., "End-Host Mobility and Multi-Homing with Host Identity Protocol", draft-nikander-hip-mm-02 (work in progress), July 2004.

[17] Nordmark, E., "Multihoming without IP Identifiers", draft-nordmark-multi6-noid-02 (work in progress), July 2004.

[18] Nordmark, E., "Multihoming using 64-bit Crypto-based IDs", draft-nordmark-multi6-cb64-00 (work in progress), November 2003.

[19] Nordmark, E., "Strong Identity Multihoming using 128 bit Identifiers (SIM/CBID128)", draft-nordmark-multi6-sim-01 (work in progress), October 2003.

[20] Vogt, C., Arkko, J., Bless, R., Doll, M. and T. Kuefner, "Credit-Based Authorization for Mobile IPv6 Early Binding Updates", draft-vogt-mipv6-credit-based-authorization-00 (work in progress), May 2004.

[21] Ylitalo, J., "Weak Identifier Multihoming Protocol (WIMP)", draft-ylitalo-multi6-wimp-01 (work in progress), July 2004.

[22] Aura, T., Roe, M. and J. Arkko, "Security of Internet Location Management", In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA., December 2002.

Author's Address

Jari Arkko
Ericsson
Jorvas 02420
Finland

EMail: jari.arkko@ericsson.com

Appendix A. Contributors

This draft attempts to summarize the thoughts and unpublished contributions of many people, including the MULTI6 WG design team members Marcelo Bagnulo Braun, Iljitsch van Beijnum, Erik Nordmark, Geoff Huston, Margaret Wasserman, and Jukka Ylitalo, the MOBIKE WG contributors Pasi Eronen, Tero Kivinen, Francis Dupont, Spencer Dawkins, and James Kempf, and my colleague Pekka Nikander at Ericsson. This draft is also in debt to work done in the context of SCTP [9].

The protocol design in Section 5.4 is due to Erik, Marcelo, and Iljitsch.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in

this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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