

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 9, 2010

M. Bagnulo  
UC3M  
J. Halpern  
Ericsson  
March 8, 2010

Analysis of data-triggered binding creation in SAVI  
draft-bagnulo-savi-analysis-02

## Abstract

The goal of this document is to serve as input to the design of the Source Address Validation architecture being defined in the SAVI WG. In particular, it analyses the different ways to handle data packets for which no binding exists, and the impact of the different approaches in the overall performance of the network.

## Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 9, 2010.

## Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

Internet-Draft

SAVI Analysis

March 2010

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

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	The Neighbour Discovery case . . . . .	<a href="#">3</a>
2.1.	Arguments against option 1: treat packets as non compliant packets . . . . .	<a href="#">4</a>
<a href="#">2.1.1.</a>	Lack of binding state due to packet loss . . . . .	<a href="#">4</a>
2.1.2.	Lack of binding state due to a change in the topology . . . . .	<a href="#">7</a>
<a href="#">2.1.3.</a>	Lack of binding state due to state loss . . . . .	<a href="#">7</a>
2.2.	Arguments against option 2: trigger the process of binding creation . . . . .	<a href="#">10</a>
<a href="#">3.</a>	The DHCP case . . . . .	<a href="#">11</a>
3.1.	Potential issues with an approach that treats packets as non compliant packets . . . . .	<a href="#">11</a>
<a href="#">4.</a>	Acknowledgments . . . . .	<a href="#">14</a>
<a href="#">5.</a>	Informative References . . . . .	<a href="#">14</a>
	Authors' Addresses . . . . .	<a href="#">14</a>

Internet-Draft

SAVI Analysis

March 2010

## 1. Introduction

The SAVI WG is chartered to produce a solution for address validation with local scope. The basic idea in SAVI is to include some SAVI devices in the topology that would enforce the proper usage of the source IP addresses contained in the packets. A major constraint in SAVI design is that SAVI must not require any changes to end hosts. This basically implies that the SAVI enforcers need to be able to determine which host is authorized to use which IP address. The proposed approaches for SAVI all concur that the SAVI device should sniff the control packets that are related to address assignment, in particular, DHCP and ND. By sniffing those packets the SAVI device can discover which host is legitimately using which address and create a binding for that address. The existence of a binding in a SAVI device implies that the SAVI device has information of which node is authorized to use the address contained in the binding, and any packet contained that address that is coming from a different point of the topology will be treated as a non-compliant packet (e.g. discarded). One aspect where there is still ongoing debate is how to handle data packets for which there is no binding. The main question here is whether to treat as a compliant packet or a non-compliant one. There are many tradeoffs involved in that design choice. The goal of this note is to explore the tradeoffs and serve as input to the ongoing debate.

Disclaimer: Joel has not able to fully review the final version and Marcelo have added some potentially controversial text in the new version, so you know who is to blame.

## 2. The Neighbour Discovery case

In the case of Neighbour Discovery (ND), the messages that are used to create bindings in the SAVI device are the Neighbour Solicitation (NSOL) and potentially the Neighbour Advertisement (NADV) that are exchanged during the Duplicate Address Detection (DAD) procedure.

Each node that configures an IP address performs the DAD procedure by sending a NSOL for the address it is about to configure in its interface. If no NADV is received, the address is assumed to be unused and it is configured in the interface. In terms of SAVI, we have mentioned that the SAVI device will create a binding when it observes a successful DAD procedure for a given address, binding the address for the DAD procedure was executed to the lower layer anchor used by the node performing the DAD.

The question that we need to address is: what does the SAVI device should do with data packets for which it has no binding information i.e. addresses for which the SAVI device has not observed a DAD NSOL

message? The possible options are:

1. Treat the packet as a non compliant packet (which in most of the cases means to discard it)
2. Trigger the process of creating a binding (whatever that is). Eventually, if the binding is successfully created, data packets coming from that lower layer anchor will be compliant and hence forwarded.

We will next consider the impact of the above options in the design of a SAVI solution.

## [2.1.](#) Arguments against option 1: treat packets as non compliant packets

The main argument against this approach is the overall robustness of the resulting network. The main concern that has been stated is that a network running SAVI that implements this option may end up disconnecting legitimate users from the network, by filtering packets coming from them. The net result would be a degraded robustness of the network as a whole, since legitimate users would perceive this as a network failure. There are three different causes that resulted in the lack of state in the binding device for a legitimate address, namely, packet loss, state loss and topology change. We will next perform an analysis for each of them.

### [2.1.1.](#) Lack of binding state due to packet loss

The DAD procedure is inherently unreliable. It consists on sending a NSOL packet and if no NADV packet is received back, success is assumed and the host starts using the address. In general, the lack

of response is because no other host has that particular address configured in their interface, but it may also be the case that the NSOL packet or the NADV packet has been lost. From the sending host perspective there is no difference and the host assumes that it can use the address. In other words, the default action is to allow the host to obtain network connectivity.

It should be noted that the loss of a DAD packet has little impact on the network performance, since address coalition is very rare and the host assumes success in that case. By designing a SAVI solution that would discard packets for which there is no binding, we are diametrically changing the default behavior in this respect, since the default would be that if the DAD packets are lost, then the node is disconnected from the network (as its packets are filtered). What is worse, the node has little clue of what is going wrong, since it has successfully configured an address but it has no connectivity. The net result is that the overall reliability of the network has significantly decreased as the loss of a single packet would imply that a host is disconnected from the network.

The only mechanism that the DAD has to improve its reliability is to send multiple NSOL. However, current [RFC4862](#) defines a default value of 1 NSOL message for the DAD procedure, so requiring any higher value would imply manual configuration of all the hosts connected to the SAVI domain.

#### [2.1.1.1](#). Why initial packets may be (frequently) lost

##### The case of LANs

Devices connecting to a network may experience periods of packet loss after the link-layer becomes available for two reasons: Invalid Authentication state and incomplete topology assessment. In both cases, physical-layer connection occurs initially and presents a medium where packets are transmissible, but frame forwarding is not available across the LAN.

For the authentication system, devices on a controlled port are forced to complete 802.1X authentication which may take multiple round trips and many milliseconds to complete (see IEEE 802.1X-2004). In this time, initial DHCP, IPv6 Neighbour Discovery, Multicast Listener or Duplicate Address Detection messages may be transmitted.

However, it has also been noted that some devices have the ability for the IP stack to not see the port as up until 802.1x has completed. Hence, that issue needs investigation to determine how common it is now.

Additionally, any system which requires user input at this stage can extend the authentication time, and thus the outage. This is problematic where hosts relying upon DHCP for address configuration time out.

Upon completion of authentication, it is feasible to signal upper layer protocols as to LAN forwarding availability. This is not typical today, so it is necessary to assume that protocols are not aware of the preceding loss period.

For environments which do not require authentication, addition of a new link can cause loops where LAN frames are forwarded continually. In order to prevent loops, all LANs today run a spanning-tree protocol, which selectively disables redundant ports. Devices which perform spanning-tree calculations are either traditional Spanning-Tree Protocol (STP) (see IEEE802.1D-1998) or rapidly converging versions of the same (RSTP/MSTP) (see IEEE 802.1D-2004 and IEEE 802.1Q-2005).

Until a port is determined to be an edge port (RSTP/MSTP), the rapid protocol speaker has identified its position within the spanning-tree

(RSTP/MSTP) or completed a Listening phase (STP), its packets are discarded.

For ports which are not connected to rapid protocol switches, it takes a minimum three seconds to perform edge port determination (see IEEE 802.1D-2004). Alternatively completion of Listening phase takes 15 seconds (see IEEE 802.1D-1998). This means that during this period, the link-layer appears available, but initial packet transmissions into and out of this port will fail.

It is possible to pre-assess ports as edge ports using manual configuration of all the involved devices and thus make them immediately transmissible. This is never default behaviour though.

The case fixed access networks

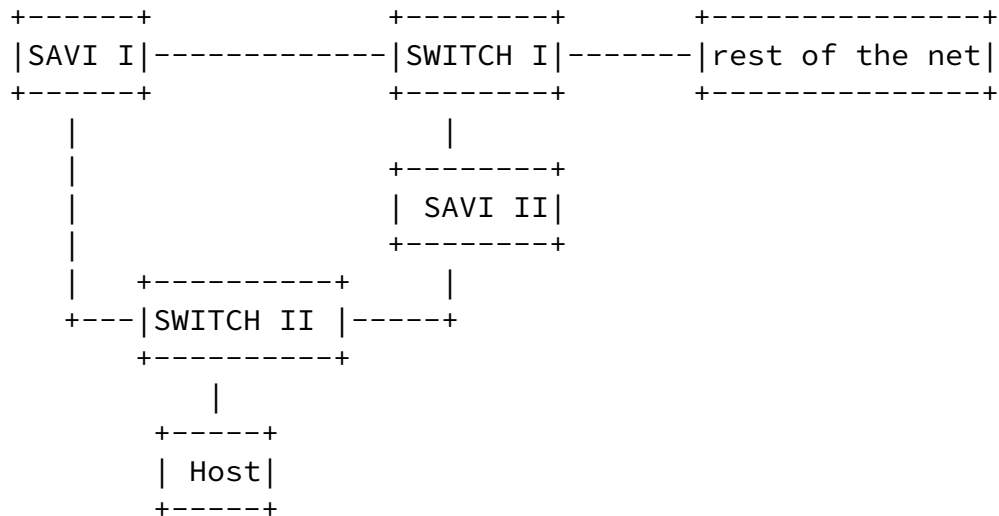
In fixed access networks such as DSL and Cable the end hosts are usually connected to the access network through a residential gateway (RG). If the host interface is initialized prior to the residential gateway getting authenticated and connected to the access network, the access network is not aware of the DAD packets that the host sent out. As an example, in DSL networks the Access Node(DSLAM) that needs to create and maintain binding state will never see the DAD message that is required to create such state.

#### 2.1.1.1.1. Special sub-case:SAVI device rate-limiting packets

A particular sub-case is the one where the SAVI device itself "drops" ND packets. In order to protect itself against DoS attacks and flash-crowds, the SAVI device will have to rate-limit the processing of packets triggering the state creation process (which require processing from the SAVI device). This implies that the SAVI device may not process all the ND packets in case it is under heavy conditions. The result is that the SAVI device will fail to create a binding for a given DAD NSOL packet, which implies that the data packets coming from the host that sent the DAD NSOL packet will be filtered if this approach is adopted. The problem is that the host will assume that the DAD procedure was successful and will not perform the DAD procedure again which in turn will imply that the host will be disconnected from the network. While it is true that the SAVI device will also have to rate limit the processing of the data packets, the host will keep on sending data packets, so it is possible to recover from the alternative approach where data packets trigger the binding creation procedure.

#### 2.1.2. Lack of binding state due to a change in the topology

In the case SAVI is being deployed in a switched Ethernet network, topology changes may result in a SAVI device receiving packets from a legitimate user for which the SAVI device does not have a binding for. Consider the following example:



Suppose that after bootstrapping all the elements are working properly and the spanning tree is rooted in the router and it includes one branch that goes SWITCH I-SAVI I- SWITCH II and another branch that goes SWITCH I-SAVI II.

Suppose that the Host boots at this moment and sends the DAD NSOL. The message is propagated through the spanning tree and it received by SAVI I but not by SAVI II. SAVI I creates the binding.

Suppose that SAVI I fails and the spanning tree reconverges to SWITCH I- SAVI II- SWITCH II. Now data packets coming from the Host will be coursed through SAVI II which does not have binding state and will drop the packets.

### [2.1.3.](#) Lack of binding state due to state loss

The other reason why a SAVI device may not have state for a legitimate address is simply because it lost it. State can be lost due to a reboot of the SAVI device or other reasons such as memory corruption. So, the situation would be as follows: The host performs the DAD procedure and the SAVI device creates a binding for the host's address. The host successfully communicate for a while. The SAVI device reboots and lost the binding state. The packets coming

from the host are now discarded as there is no binding state for that



address. It should be noted that in this case, the host has been able to use the address successfully for a certain period of time.

Architecturally, the degradation of the network robustness in this case can be easily explained by observing that this approach to SAVI implementation breaks the fate-sharing principle. [RFC 1958](#) reads:

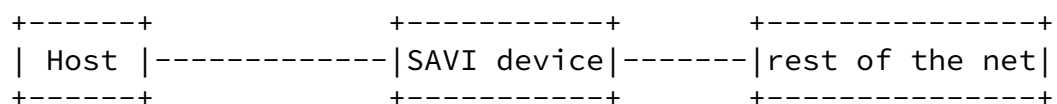
An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks (known as fate-sharing).

By binding the fate of the host's connectivity to the state in the SAVI device, we are breaking this principle and the result is degraded network resilience.

Moving on to more practical matters, we can dig deeper into the actual behaviour by considering two scenarios, namely, the case where the host is directly connected to the SAVI device and the case where there is an intermediate device between the two.

#### [2.1.3.1](#). The case of a host directly connected to the SAVI device

The considered scenario is depicted in the following picture:



The key distinguishing element of this scenario is that the host is directly connected to the SAVI device. As a result, if the SAVI device reboots, the host will see the carrier disappear and appear again.

[RFC4862](#) requires that the DAD procedure is performed when the IP address is assigned to the interface, quoting [RFC4862 section 5.4](#). Duplicate Address Detection:

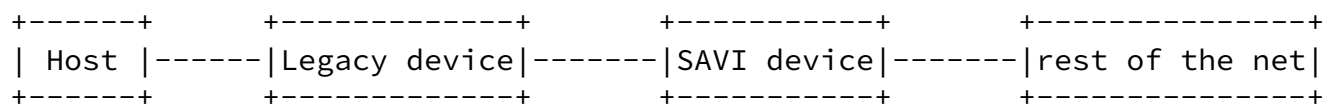
Duplicate Address Detection MUST be performed on all unicast addresses prior to assigning them to an interface, regardless of whether they are obtained through stateless autoconfiguration, DHCPv6, or manual configuration, with the following exceptions:...

However, it has been stated that some of the widely used OSes

actually do perform DAD each time the link is up, but further data would be required to take this for granted. Assuming that behaviour, that implies that if the loss of state in the SAVI device also results in the link to the host going down, then the host using the tested OSES would redo the DAD procedure allowing the recreation of the binding state in the SAVI device and preserving the connectivity of the host. This would be the case if the SAVI device reboots. It should be noted though, that it is also possible that the binding state is lost for whatever error in the SAVI process and that the SAVI link does not go down. In this case, the host would not redo the DAD procedure. However, it has been pointed out that it would be possible to require the SAVI process to flap the links of the device it is running, in order to make sure that the links go down each time the SAVI process restarts and improving the chances the host will redo the DAD procedure when the SAVI process is rebooted.

[2.1.3.2](#). The case of a host connected to the SAVI device through one or more legacy devices.

The considered scenario is depicted in the following picture:



The key distinguishing element of this scenario is that the host is not directly connected to the SAVI device. As a result, if the SAVI device reboots, the host will not see any changes.

In this case, the host would get disconnected from the rest of the network since the SAVI device would filter all its packets once the state has gone. As the node will not perform the DAD procedure again, it will remain disconnected until it reboots.

As a final comment, it should be noted that it may not be obvious to the network admin which scenario its network is running. Consider the case of a campus network where all the switches in the network are SAVI capable. A small hub connected in the office would turn this into the scenario where the host is not directly connected to the SAVI device. Moreover, consider the case of a host running multiple virtual machines connected through a virtual hub, depending on the implementation of such a virtual hub, may turn a directly

connected host scenario to the scenario where the multiple (virtual) hosts are connected through a legacy (virtual) hub.

#### [2.1.3.2.1.](#) Enforcing direct connectivity between the SAVI device and the host

Some people have argued that enforcing the direct connectivity between the SAVI device and the end host is actually a feature.

There are several comments that can be made in this respect:

First, it may well be the case in some scenarios this is desirable, but it is certainly not the case in most scenarios. Because of that, the issue of enforcing direct connectivity must be treated as orthogonal to how data packets for which there is no binding are treated, since a general solution must support directly connected nodes and nodes connected through legacy switches.

Second, as a matter of fact, the resulting behaviour described above would not actually enforce direct connectivity between the end host and the SAVI device as it would work as long as the SAVI device would not reboot. So, the argument being made is that this approach is not good enough to provide a robust network service, but it is not bad enough to enforce the direct connectivity of host to the SAVI switch.

Third, it should be noted that topology enforcement is not part of the SAVI problem space and that the SAVI problem by itself is hard enough to add additional requirements.

#### [2.2.](#) Arguments against option 2: trigger the process of binding creation

The main argument against the option of using data packets for which there is no binding to trigger the binding creation process is as follows:

It has been stated that some switch architectures would not be able to implement a SAVI solution that triggers complex actions based on data packets. The argument is that some architectures may be able to perform simple actions such as forward or discard, but they wouldn't be able to do more complex actions, such as triggering the binding creation process, that would likely imply sending some packets and creating the binding internally. It has been accepted though, that some switch architectures would be able to trigger the binding creation procedure upon the reception of a

data packet. So, if a solution would rely on triggering the binding creation as the result of receiving a data packet, it seems to be the case that some implementations would not be able to comply with the resulting RFC while some other implementations would.

Another argument against this option has to do with the added complexity. It is obvious that since this approach is a superset of the previous one it is more complex. In particular, since the SAVI

device needs to react upon data packets, it would require more processing power than the alternative approach. (this requires more elaboration)

NOTE: Is there any other argument against this option?

### [3.](#) The DHCP case

Similar to the ND case, the DHCP based SAVI will create a binding state after observing the message exchange that results in a successful IP address assignment from the DHCP server to the host. The question is what to do with data packets for which there is no binding state. Similar to the ND case, the options are either to treat it as a non compliant packet (i.e. drop) or to trigger the binding creation procedure. However, at the time of this writing, only the details of a solution that treat these packets as non compliant have been fleshed out and it is not clear how a solution that triggers the binding creation would work. So, in this section, we mostly point out some issues that may require some thought when considering a solution that treats data packets for which there is no binding as non compliant packets and the impact that such solution could have on the overall performance of the network.

The DHCP case is different than the ND case, for two main reasons:

- o The DHCP exchange is reliable and in case of failure the node does not acquire an address (as opposed to the ND case, which is unreliable and that in case of DAD packet getting lost, the host does acquire the address).
- o Some of the address assignment information is stored in the DHCP server, so in case of failure, there is a central repository to retrieve some of the information.

### 3.1. Potential issues with an approach that treats packets as non compliant packets

Since the DHCP exchange is reliable, the arguments based on packet loss do not apply to the DHCP case. We then only need to consider the arguments based on state loss and on topology change.

Arguments based on state loss

We need to distinguish the two topologies analyzed in the ND case, namely, the case of the host directly connected to the SAVI device and the case of the host connected to a legacy device.

In the case the host is directly connected to the SAVI device, the DHCP specification ([RFC3315](#)) reads:

Whenever a client may have moved to a new link, the prefixes from the addresses assigned to the interfaces on that link may no longer be appropriate for the link to which the client is attached. Examples of times when a client may have moved to a new link include:

- \* The client reboots.
- \* The client is physically connected to a wired connection.
- \* The client returns from sleep mode.
- \* The client using a wireless technology changes access points.

In any situation when a client may have moved to a new link, the client MUST initiate a Confirm/Reply message exchange.

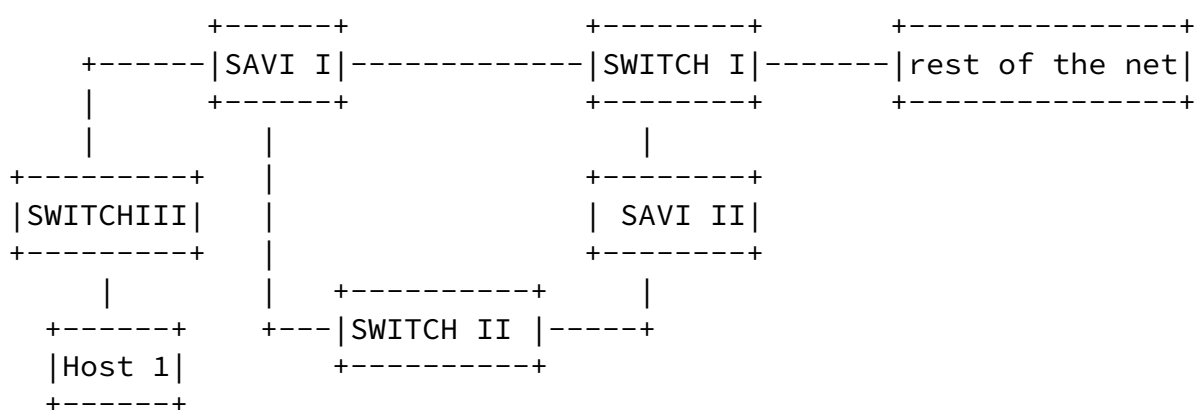
It is not clear what should a host do when for instance the AP that it is directly connected to reboots. In this case, the access point is still the same, so there is not clear guidance on [RFC3315](#). there are claims that some OSes do redo the Confirm/Reply exchange when the link flaps, but further data would be required to take this for granted.

Assuming that behaviour, we could rely on the same trick described earlier, about requiring the SAVI process to flap all the links of the device it is running on, in order to deal with SAVI process failure modes that do not imply a reboot of the whole device.

In the case the host is connected to a legacy device, the rebooting

of the SAVI device would not result in the host performing a Confirm/Reply exchange. In this case, it is not clear (to us at least) how the SAVI device could restore the lost SAVI binding state. One option could be to try to retrieve it from the DHCP server. One potential problem with that is that it is not obvious that the DHCP server knows the lower layer anchor information. In addition, that would require a protocol between the SAVI device and the DHCP server. Another possible option would be that the SAVI device forces the host to perform a Confirm/Reply exchange, but since they are not directly connected, it is not obvious how this could be done.

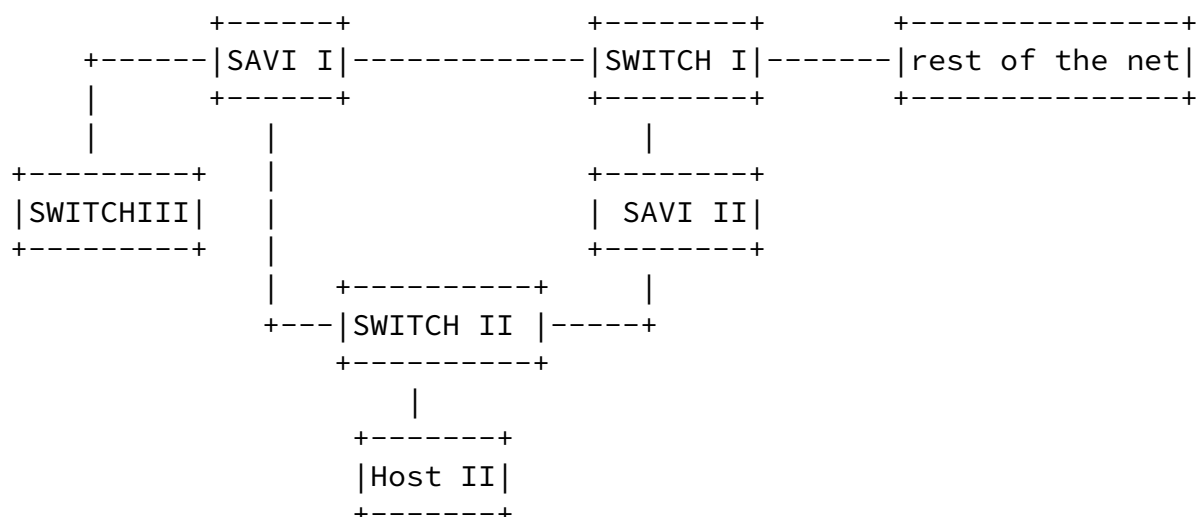
It has been argued that this problem can be solved if the binding information is stored in non-volatile memory. Staying away from the implementation aspect of whether this is feasible in the different switch architectures, the problem with non-volatile memory to store dynamic information is that they are, actually non volatile. This may result in the SAVI device filtering based on stalled information. Consider the following case:



Consider the case where we have SAVI I and SAVI II storing the SAVI state in non volatile memory. Suppose that Host I connects to the network and gets IP address IP1 from the DHCP server.

Suppose that now the power of SAVI I goes down and stays down for a

few hours. Suppose that Host I leaves the network and Host II attaches to the network in SWITCH II as depicted below. The spanning tree goes SWITCH I-SAVI II-SWITCH II.



Suppose that SAVI I boots and now the spanning tree changes and goes SWITCH I- SAVI I- SWITCH II. Now packets of Host II will be forwarded through SAVI I. But Host II has IP1 as IP address, but SAVI I still holds the state referring to Host I through the port through which SWITCH III is connected. The result is that SAVI I will drop the packets coming from Host II.

Arguments based on change in the topology

the same argument referring to changes in the topology apply to the DHCP case. The result is that changes in the topology may result in the SAVI device filtering packets for legitimate users.

#### [4.](#) Acknowledgments

Greg Daley and Surech Krishnan provided the text for section entitled "Why initial packets may be (frequently) lost"

Alberto Garcia Martinez brought up the issues related to change in

the topology and provided the topologies for both the cases of topology change as well as the case for non volatile memory.

Marcelo Bagnulo is partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program and by the Telefonica Chair at University Carlos III of Madrid..

## 5. Informative References

- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", [RFC 4862](#), September 2007.
- [RFC1958] Carpenter, B., "Architectural Principles of the Internet", [RFC 1958](#), June 1996.

## Authors' Addresses

Marcelo Bagnulo  
Universidad Carlos III de Madrid  
Av. Universidad 30  
Leganes, Madrid 28911  
SPAIN

Phone: 34 91 6248814  
Email: [marcelo@it.uc3m.es](mailto:marcelo@it.uc3m.es)  
URI: <http://www.it.uc3m.es>

Joel M. Halpern  
Ericsson

Phone: 1 703 371 3043  
Email: [joel.halpern@ericsson.com](mailto:joel.halpern@ericsson.com)