SAVI design

marcelo bagnulo & Joel Halpern

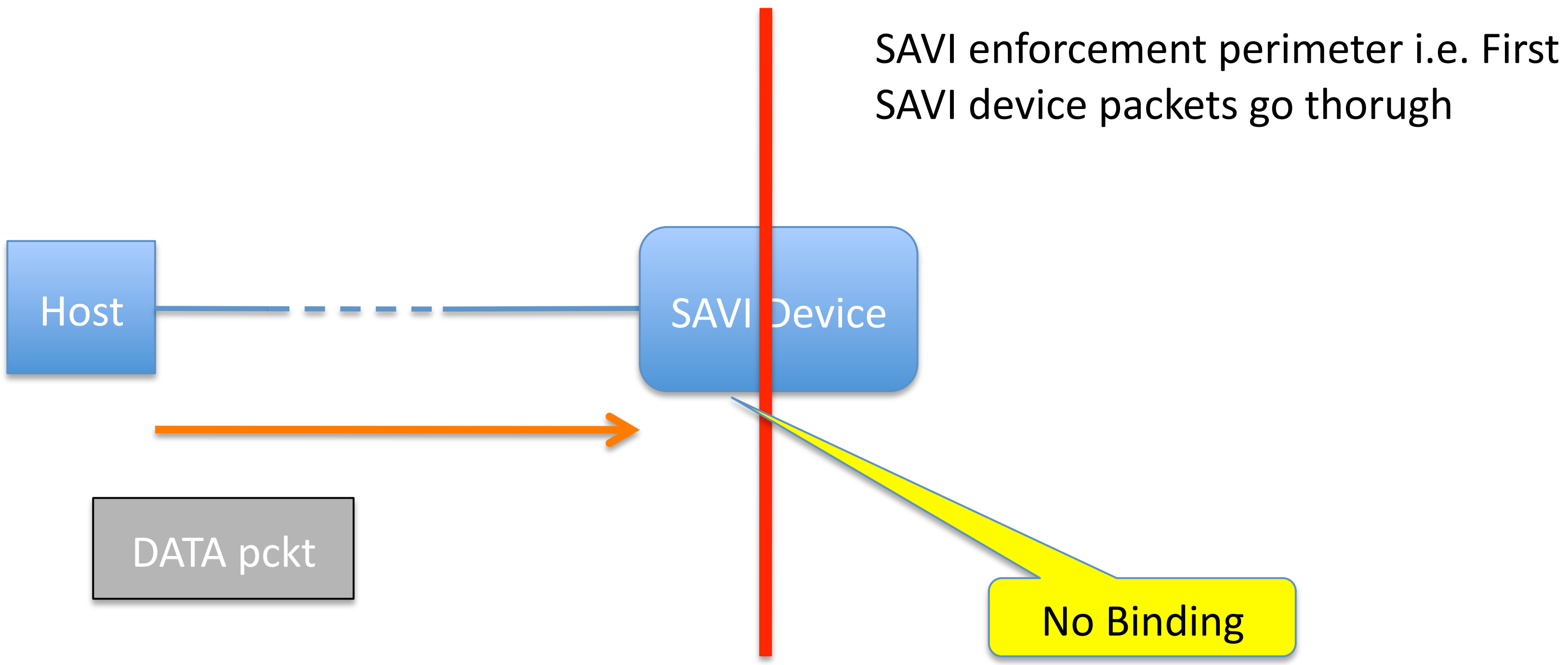
IETF77

Some design choices made

- SAVI bindings will be created upon the reception of control packets
  - DAD ND for SLAAC
  - DHCP for DHCP-SAVI
- SAVI enforcement perimeter
  - Only the SAVI device that is first reached will store a binding

Open design question:

What should a SAVI device do when it receives a ***data packet*** for with it has ***no binding***?


SAVI enforcement perimeter i.e. First SAVI device packets go thorugh
Host
SAVI Device
DATA pckt
No Binding

Options:
-Drop the packet
-Trigger the binding creation procedure

Issues covered

- In this presentations we will consider
  - Arguments against each option
  - In depth analysis of SLAAC case
  - Preliminary analisys of the DHCP case
  - Scenario where the host is directly connected to the SAVI device
  - Scenario where the host is connected to the SAVI device through a legacy switch

SLAAC case

- Arguments against dropping data packets for which there is no binding:
  - Overall network reduced performance
  - Opens the possibility of dropping packets from legitimate users
  - Perceived as a network failure

SLAAC case

- The SLAAC-SAVI device creates binding upon the completion of the DAD process.
- Is it possible that a SLAAC SAVI device receives a packet from a legitimate host for which it doesn’t have a binding?
  – Answer: Yes
- And the reasons why are:
  – Lack of binding state due to packet loss
  – Lack of binding state due to SAVI state loss
  – Lack of binding state due to topology change

SLAAC case: Packet loss

- DAD process in inherently unreliable
- Host sends a NSOL and waits.
  - No answer means sucess!
    - The host starts using the address
  - Indistinguishable from the case the NSOL is lost
- In case the NSOL packet was lost:
  - The host will assume sucess, but,
  - The SAVI device will drop the packets

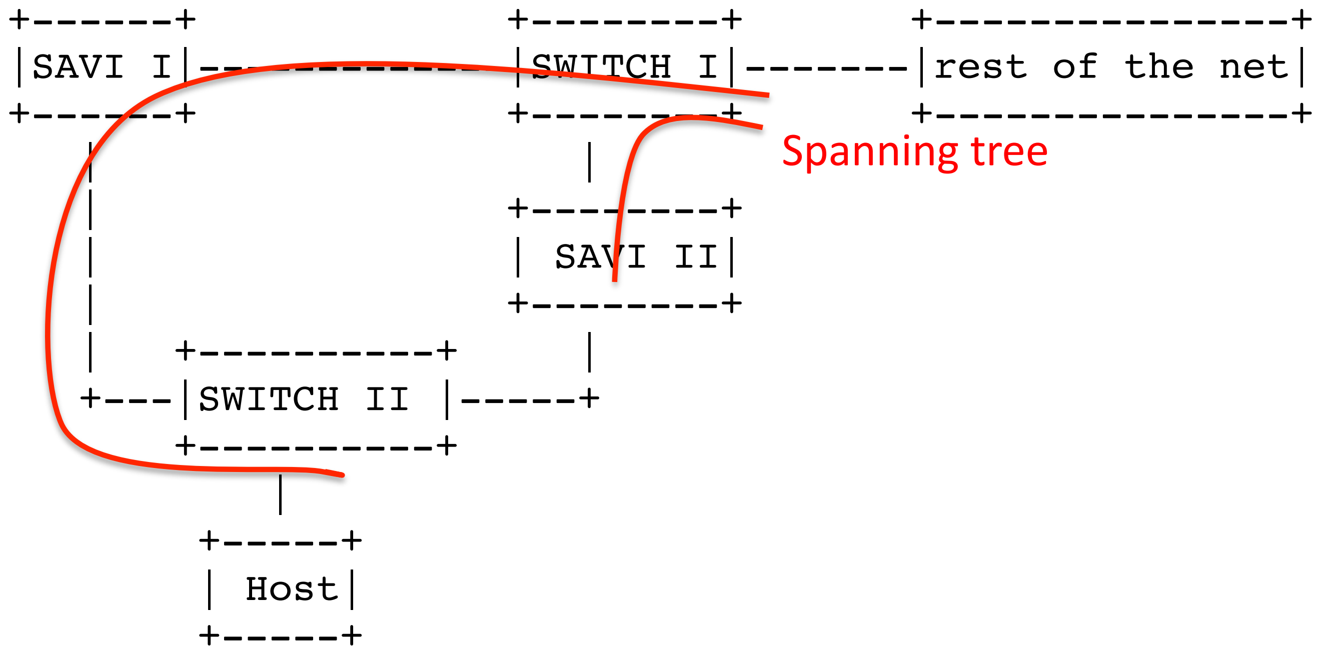
SLAAC case: packet loss (II)

- Additional causes for DAD NSOL packets loss
  - DAD NSOL is the first packet a host sends
  - Initial packets can potentially suffer a higher loss rate due to
    - Authentication delay
    - Spanning tree formation
  - Rate limiting: SAVI device will protect against DoS attacks by ***rate-limiting*** the packets that result in processing.
    - This implies that potentially some DAD NSOL msgs may NOT be processed by the SAVI device

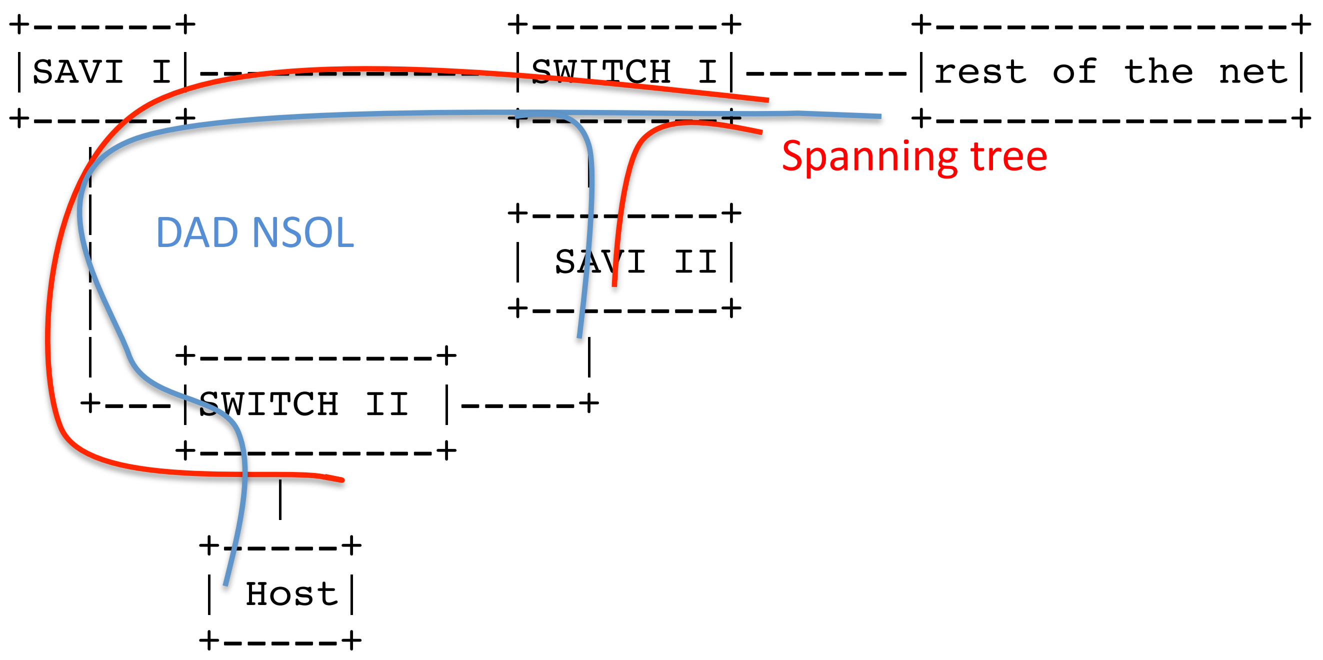
SLAAC case: topology change

```
+-------+            +--------+       +---------------+
|SAVI I |------------|SWITCH I|-------|rest of the net|
+-------+            +--------+       +---------------+
   |                    |
   |                 +--------+
   |                 | SAVI II|
   |                 +--------+
   |   +----------+     |
   +---|SWITCH II |-----+
       +----------+
           |
        +-----+
        | Host|
        +-----+
```

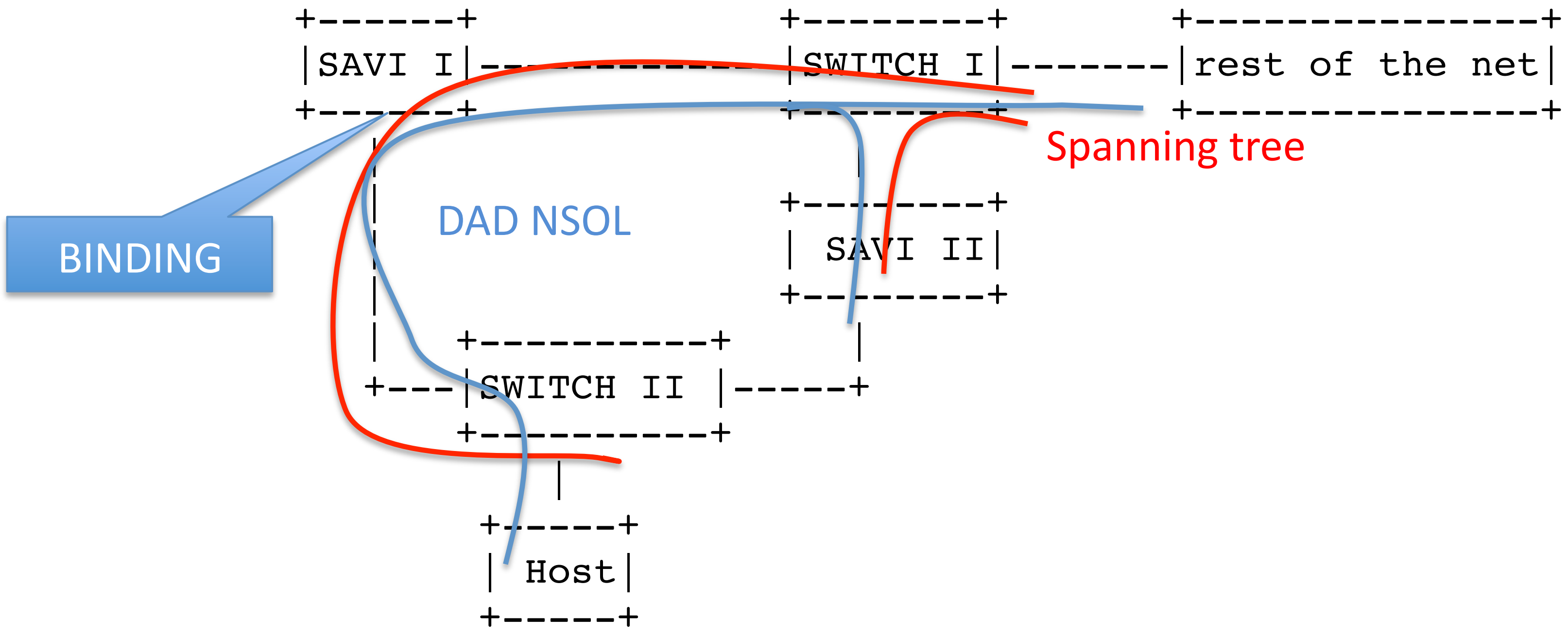
SLAAC case: topology change



SLAAC case: topology change

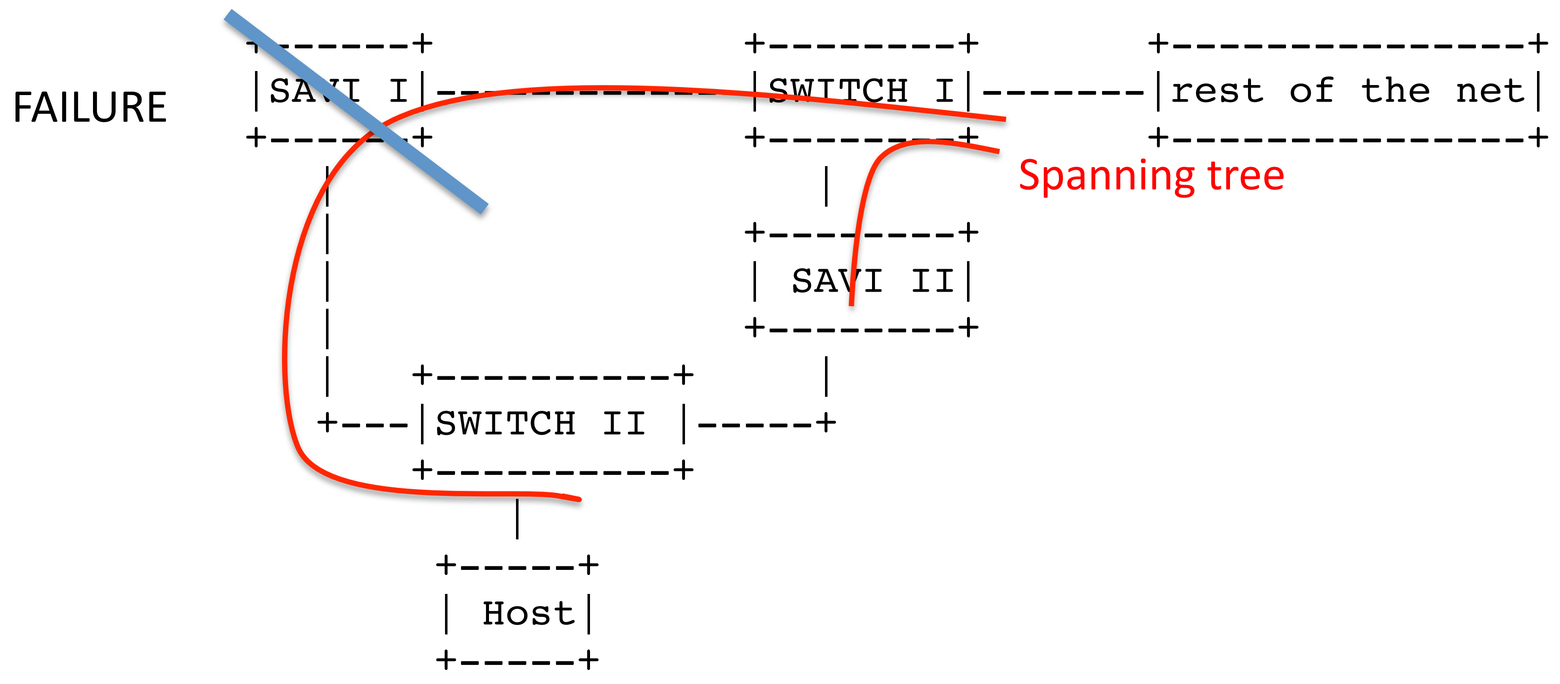


SLAAC case: topology change

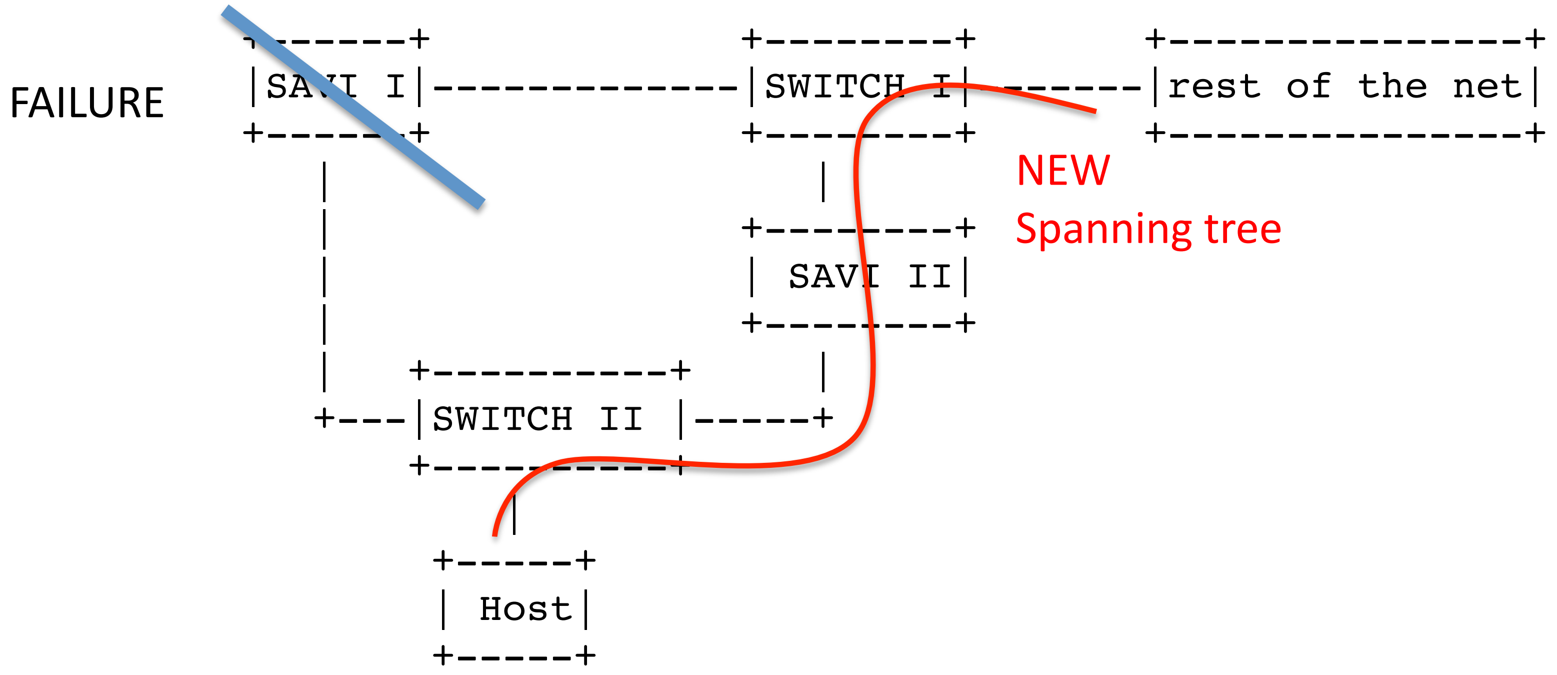


```
+-------+           +---------+       +---------------+
|SAVI I|------------|SWITCH I|-------|rest of the net|
+-------+           +---------+       +---------------+
   |                     |
   |                +---------+
   |                | SAVI II|
   |                +---------+
   |   +-----------+     |
   +---|SWITCH II  |-----+
       +-----------+
             |
         +------+
         | Host|
         +------+
```

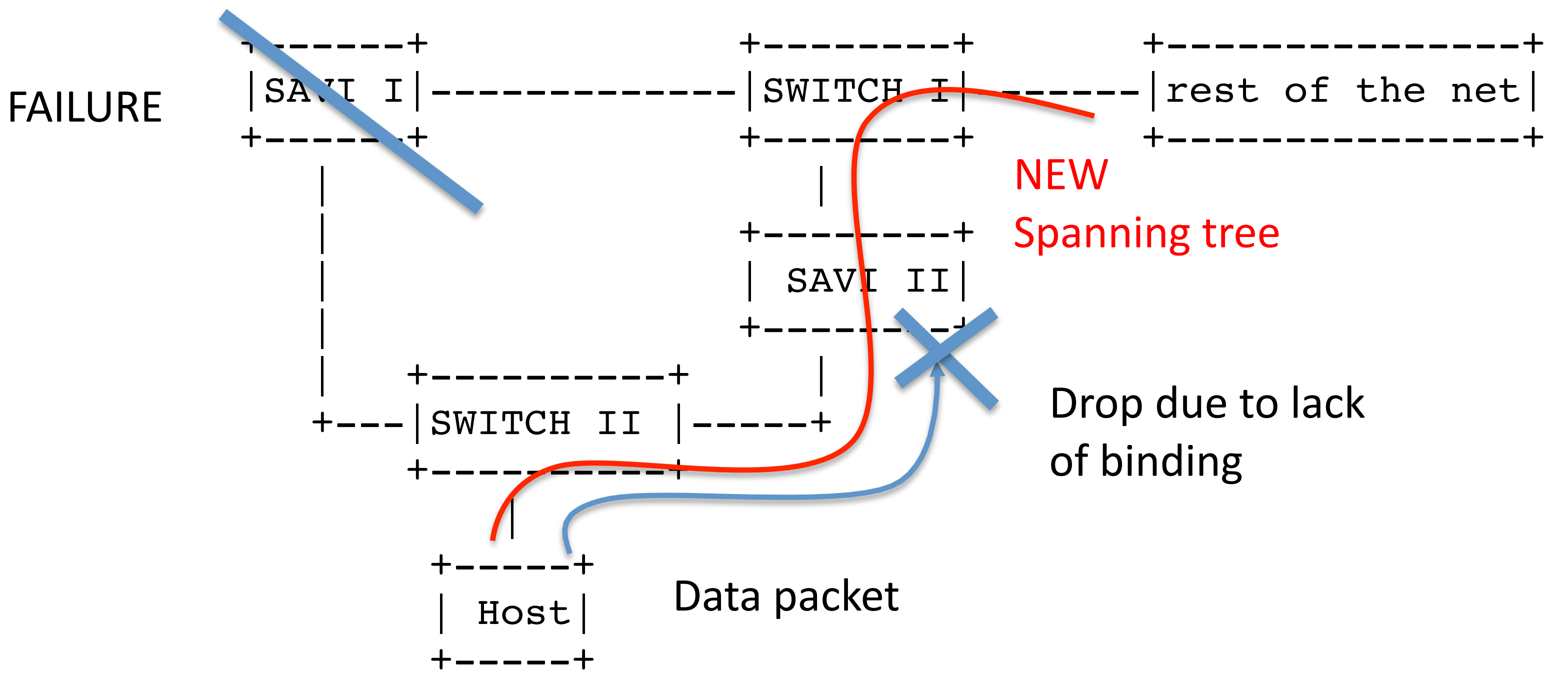
SLAAC case: topology change



SLAAC case: topology change



SLAAC case: topology change



SLAAC case: lost state

- Problem case:
  1) Host perform DAD,
  2) SAVI state is created
  3) Hosts communicates succesfully
  4) SAVI device reboots
  5) Following packets are discarded by SAVI device

SLAAC case: lost state

- Architectural cause:
  - 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).

SLAAC case: lost state Directly connection SAVI-host

```
+-------+       +------------+      +----------------+
| Host  |-------|SAVI device|-------|rest of the net|
+-------+       +------------+      +----------------+
```

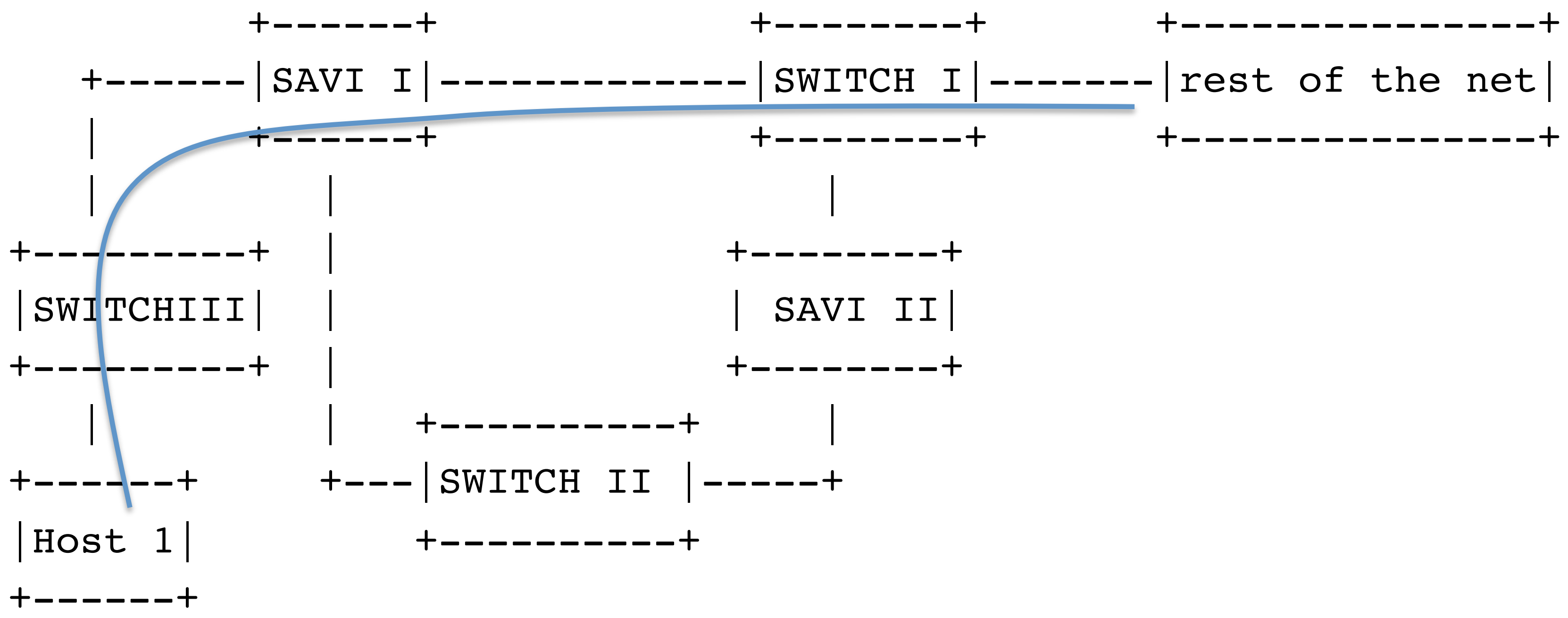
- RFC4862 section 5.4. Duplicate Address Detection: Duplicate Address Detection MUST be performed on all unicast addresses prior to assigning them to an interface, regardless of whether they are obtained through stateless autoconfiguration, DHCPv6, or manual configuration,
- While not required by RFC4862, it seems that some widely used OSes do DAD when the link is down and up.
  - More experiments would be needed to check every platform

SLAAC case: lost state
Connection through a legacy switch

```
+------+    +-------------+     +-----------+    +---------------+
| Host |----|Legacy device|-----|SAVI device|----|rest of the net|
+------+    +-------------+     +-----------+    +---------------+
```

- In this case, the host will NOT redo the DAD, since the link flap is invisible to the host.
- A possible appraoch to deal with this, would be to require the SAVI state to be stored in non volatile memory
  - Doesn't solve all the problems, see next slide

SLAAC case: lost state
Legacy switches/non volatile memory



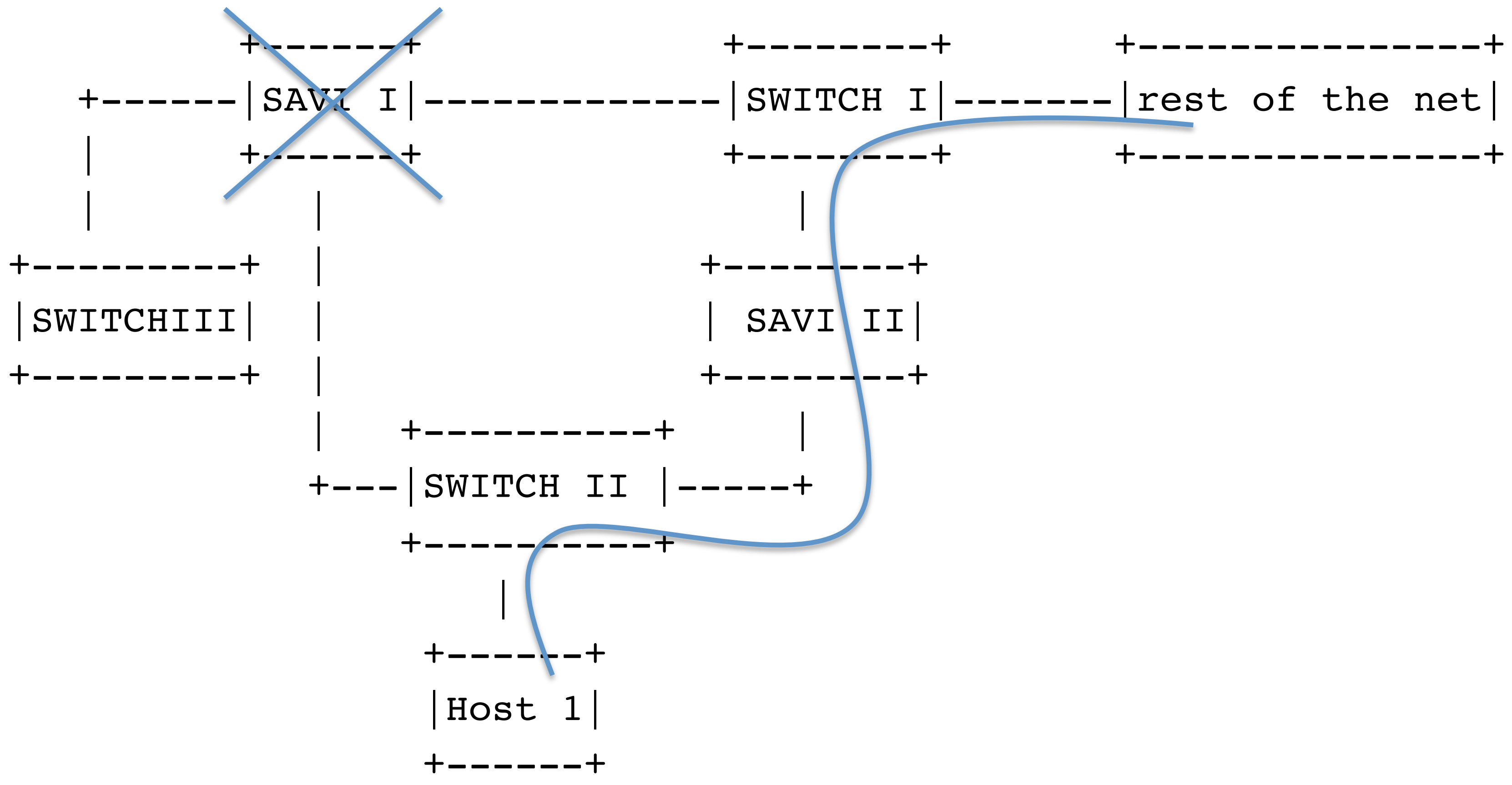
SLAAC case: lost state
Legacy switches/non volatile memory

```
          +-------+                   +---------+      +----------------+
   +------|SAVI  I|-------------------|SWITCH I|-------|rest of the net|
   |      +-------+                   +---------+      +----------------+
   |          |                           |
+---------+   |                         +---------+
|SWITCHIII|   |                         | SAVI II|
+---------+   |                         +---------+
   |          |   +-----------+             |
+-------+     +---|SWITCH II  |-------------+
|Host 1|          +-----------+
+-------+
```

SLAAC case: lost state
Legacy switches/non volatile memory

```
          +------+             +--------+       +---------------+
   +------|SAVI I|-------------|SWITCH I|-------|rest of the net|
   |      +------+             +--------+       +---------------+
   |         |                    |
+---------+  |                +--------+
|SWITCHIII|  |                | SAVI II|
+---------+  |                +--------+
             |   +----------+     |
             +---|SWITCH II |-----+
                 +----------+
                     |
                  +------+
                  |Host 1|
                  +------+
```

SLAAC case: lost state
Legacy switches/non volatile memory



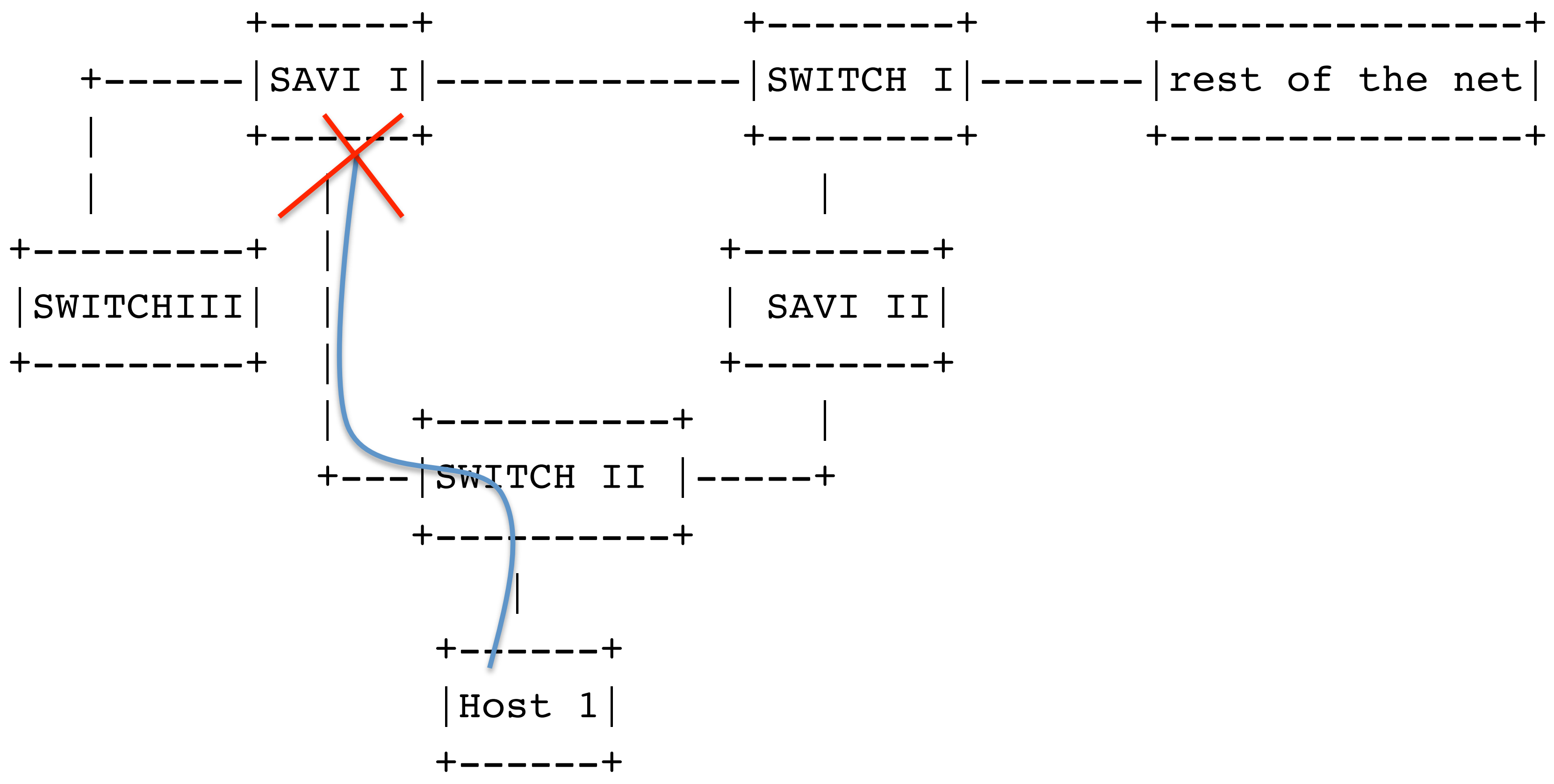
```
               +-------+                 +---------+      +----------------+
      +-------|SAVI I|-------------|SWITCH I|-------|rest of the net|
      |        +-------+                 +---------+      +----------------+
      |                |                      |
 +----------+          |                 +---------+
 |SWITCHIII|           |                 | SAVI II|
 +----------+          |                 +---------+
                       |   +-----------+      |
                       +---|SWITCH II |-----+
                           +-----------+
                                |
                            +-------+
                            |Host 1|
                            +-------+
```

SLAAC case: lost state Legacy switches/non volatile memory

```
          +------+             +--------+       +---------------+
   +------|SAVI I|-------------|SWITCH I|-------|rest of the net|
   |      +------+             +--------+       +---------------+
   |         |                    |
+---------+  |                +--------+
|SWITCHIII|  |                | SAVI II|
+---------+  |                +--------+
             |   +----------+     |
             +---|SWITCH II |-----+
                 +----------+
                     |
                  +------+
                  |Host 1|
                  +------+
```

STP reconveges
through SAVI I

SLAAC case: lost state
Legacy switches/non volatile memory



```
          +-------+                    +---------+      +----------------+
   +------|SAVI I|--------------|SWITCH I|--------|rest of the net|
   |      +-------+                    +---------+      +----------------+
   |         |                             |
+----------+ |                          +---------+
|SWITCHIII| |                          | SAVI II|
+----------+ |                          +---------+
             |    +-----------+             |
             +---|SWITCH II |-----+
                  +-----------+
                      |
                   +-------+
                   |Host 1|
                   +-------+
```

SLAAC case

- Arguments against triggering the binding creation process upon the recepetion of data packets for which there is no binding
- Some architectures (esp. low end ones) may have problems triggering actions upon the reception of data packets.
- Added complexity

DHCP case

- Different from SLAAC case: only a subset of the problems
  - DHCP exchange is reliable, lack of binding due to packet loss is not an issue
  - The lack of binding due to state loss is similar to SLAAC one
    - Can be mitigated with non volatile memory, while some topologies still have problem.
  - The lack of binding due to topology changes is also similar to the SLAAC one

Requirement level for data packet triggered binding creation

- SLAAC SAVI
  - SHOULD (properly qualified)
  - MUST (both Marcelo and Joel’s favorite)
- DHCP SAVI
  - MUST (marcelo’s favorite)
  - Qualified SHOULD + MUST store binding in non volatile memory (Joel’s favorite)